THE SCRUM ANTI-PATTERNS GUIDE

CHALLENGES EVERY SCRUM TEAM FACES AND HOW TO OVERCOME THEM

STEFAN WOLPERS

Forewords by **DAVE WEST & JANNA BASTOW**

The Professional Scrum Series by Scrum.org

Stefan Wolpers has a remarkable ability to highlight underlying traps and issues for stakeholders, teams, and process. In *The Scrum Anti-Patterns Guide*, Wolpers documents sources of waste and frustration, an amazing compendium of typical ways progress becomes blocked. Depressing! He doesn't leave us there though. He also recommends insightful remedies. Uplifting!

—*Diana Larsen, speaker, advisor, author at dianalarsen.com*

This book is an invaluable treasure for every Scrum practitioner as it shows not only what can (and does) go wrong in Scrum teams, but also how to avoid and overcome these traps. Stefan shares his in-depth experience and provides concrete tips to help the whole Scrum team grow.

—*Jutta Eckstein, coauthor of* Company-Wide Agility with Beyond Budgeting, Open Space & Sociocracy

Stefan Wolpers' comprehensive collection of common Scrum anti-patterns is a great resource for anybody who wants to improve their application of Scrum and fully leverage the framework. Highly recommended!

—*Roman Pichler, author of* Agile Product Management with Scrum

There is so much potential with Scrum that sadly ends up wasted because teams misuse the process. Wolpers does a masterful job of identifying the anti-patterns that cause this waste and points out exactly how to overcome each one. This is a must-have book for all Scrum practitioners.

—*Jeff Gothelf, author of* Lean UX *and* Sense & Respond

The first step to fixing a problem is acknowledging you have a problem. Stefan Wolpers' wonderful book is the ultimate compendium of Scrum anti-patterns. After exposing common Scrum problems, the book serves as a cunning companion that masterfully guides you to transcend the dysfunctions it lays bare.

—*Maarten Dalmijn, author of* Driving Value with Sprint Goals

Stefan's groundbreaking book *The Scrum Anti-Patterns Guide* flips the script, spotlighting Scrum pitfalls over best practices. Packed with vivid examples, clear illustrations, and robust theory, this book accelerates learning by showing you what NOT to do. Not only insightful but delightfully entertaining!

—*Jorgen Hesselberg, author of* Unlocking Agility: An Insider's Guide to Agile Enterprise Transformation *and Co-Founder of Comparative Agility*

For years I've admired Stefan Wolpers' shared anti-patterns of various roles and activities within and around Scrum teams. His insights are simple, insightful, and powerful, while always rooted in his practical experience. He also artfully balances the anti-patterns away from criticism and toward improvement. This book contains his collective wisdom and I couldn't recommend it more to anyone desiring to improve their Scrum.

—*Bob Galen, Principal Coach at Agile Moose*

The Scrum Anti-Patterns Guide

The Professional Scrum Series by Scrum.org

PRACTICAL PRODUCT MANAGEMENT FOR PRODUCT OWNERS
CREATING WINNING PRODUCTS WITH THE PROFESSIONAL PRODUCT OWNER STANCES
CHRIS LUKASSEN
ROBBIN SCHUURMAN
THE PROFESSIONAL AGILE LEADER
THE LEADER'S JOURNEY TOWARD GROWING MATURE AGILE TEAMS AND ORGANIZATIONS
RON ERINGA
KURT BITTNER
LAURENS BONNEMA
THE PROFESSIONAL SCRUM TEAM
GROWING AND EMPOWERING CROSS-FUNCTIONALITY AND RESILIENCY IN A COMPLEX WORLD
PETER GÖTZ
ZOMBIE SCRUM SURVIVAL GUIDE
A JOURNEY TO RECOVERY
CHRISTIAAN VERWIJS
JOHANNES SCHARTAU
BARRY OVEREEM

Visit **informit.com/scrumorg** for a complete list of available publications.

The **Professional Scrum Series** from Pearson Addison-Wesley and Scrum.org consists of a series of books that focus on helping individuals and organizations apply Scrum and agile leadership to improve the outcomes of customers and organizations. Approaching the challenge from different perspectives, each book provides deep insights into overcoming the obstacles that both teams and organizations face as they seek to reap the benefits of agility.

All Scrum.org proceeds from the series go to Year Up, an organization whose mission is to close the Opportunity Divide by providing urban young adults with the skills, experience, and support to empower them to reach their potential through professional careers and education.

Pearson

twitter.com/informIT

informIT.com
the trusted technology learning source

The Scrum Anti-Patterns Guide

Challenges Every Scrum Team Faces and How to Overcome Them

Stefan Wolpers

Addison-Wesley

Cover image: Bakemon/Shutterstock

FIGB-07-Ralph Jocham

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2023950677

Copyright © 2024 Pearson Education, Inc.

Hoboken, NJ

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearson.com/permissions.

ISBN-13: 978-0-13-797796-3
ISBN-10: 0-13-797796-4

$PrintCode

Pearson's Commitment to Diversity, Equity, and Inclusion

Pearson is dedicated to creating bias-free content that reflects the diversity of all learners. We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, socioeconomic status, ability, age, sexual orientation, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where:

- Everyone has an equitable and lifelong opportunity to succeed through learning.
- Our educational products and services are inclusive and represent the rich diversity of learners.
- Our educational content accurately reflects the histories and experiences of the learners we serve.
- Our educational content prompts deeper discussions with learners and motivates them to expand their own learning (and worldview).

While we work hard to present unbiased content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.

- Please contact us with concerns about any potential bias at https://www.pearson.com/report-bias.html.

This page intentionally left blank

Contents

Foreword by Dave West xv
Foreword by Janna Bastow xvii
Preface xix
Acknowledgments xxi
About the Author xxiii
Introduction xxv

Chapter 1 Scrum Master Anti-Patterns 1
Introduction 1
The Scrum Master According to the Scrum Guide 2
Possible Reasons Why Scrum Masters Leave the Path 2
Anti-Patterns from Acting as an Agile (Line) Manager 5
Scrum Master Anti-Patterns by Scrum Events 15
The Sprint Planning 15
The Sprint 17
The Daily Scrum 20
The Retrospective 22
Food for Thought 26
Conclusion 26

Chapter 2 Product Owner Anti-Patterns 29
Introduction 29
The Role of the Product Owner According to the Scrum Guide 30
Product Backlog and Refinement Anti-Patterns 31
Sprint Planning Anti-Patterns 40
Sprint Anti-Patterns 42
Product Owner Anti-Patterns during the Daily Scrum 46
Sprint Review Anti-Patterns 50
Food for Thought 52
Conclusion 52

Chapter 3 Scrum Developer Anti-Patterns 55
Introduction 55
The Role of the Developers in Scrum 56
Developer Anti-Patterns by Scrum Events 56
Sprint Anti-Patterns 56
Sprint Planning Anti-Patterns of the Developers 66
Anti-Patterns during the Daily Scrum 68
Developer Anti-Patterns Concerning the Sprint Review 75
Sprint Retrospective Anti-Patterns of Developers 76
Anti-Patterns at the Product Backlog Level 77
Food for Thought 81
Conclusion 81

Chapter 4 Scrum Stakeholder Anti-Patterns 83
Introduction 83
Common Scrum Stakeholder Anti-Patterns 84
Scrum Stakeholder Anti-Patterns at the Organizational Level 84
Sprint Anti-Patterns of the IT Management 96
Incentivized Scrum Stakeholder Anti-Patterns 98
Stakeholder Anti-Patterns at Scrum Event Level 104
Product Backlog and Refinement Anti-Patterns 105
The Daily Scrum 105
Sprint Planning Anti-Patterns of Stakeholders 106
The Sprint Review 106

The Sprint Retrospective 107
Food for Thought 108
Conclusion 108

Chapter 5 Sprint Anti-Patterns 109
Introduction 109
The Purpose of the Sprint 109
Sprint Anti-Patterns 111
Sprint Anti-Patterns of the Product Owner 111
Sprint Anti-Patterns of the Developers 112
Sprint Anti-Patterns of the Scrum Master 113
Sprint Anti-Patterns of the Scrum Team 113
Sprint Anti-Patterns of the IT Management 118
Sprint Review Anti-Patterns of Stakeholders 121
Food for Thought 123
Conclusion 124

Chapter 6 Sprint Planning Anti-Patterns 125
Introduction 125
Preparing the Sprint Planning 127
Sprint Planning Anti-Patterns 127
Sprint Planning Anti-Patterns of the Developers 127
Sprint Planning Anti-Patterns of the Product Owner 132
Sprint Planning Anti-Patterns of the Scrum Team 134
Sprint Planning Anti-Patterns of the Scrum Master 138
Food for Thought 139
Conclusion 140

Chapter 7 Daily Scrum Anti-Patterns 141
Introduction 141
The Purpose of the Daily Scrum According to the Scrum Guide 142
Daily Scrum Anti-Patterns 143
Daily Scrum Anti-Patterns of the Scrum Team 143
Daily Scrum Anti-Patterns of the Developers 146
Daily Scrum Anti-Patterns of the Product Owner 150

Daily Scrum Anti-Patterns of the Scrum Master 150
Daily Scrum Anti-Patterns of the Stakeholders 151
Food for Thought 154
Conclusion 155

Chapter 8 Sprint Review Anti-Patterns 157
Introduction 157
The Scrum Guide on the Sprint Review 158
Sprint Review Anti-Patterns 159
Sprint Review Anti-Patterns of the Scrum Team 159
Sprint Review Anti-Patterns of the Product Owner 162
Sprint Review Anti-Patterns of the Developers 165
Sprint Review Anti-Patterns of the Stakeholders 167
Food for Thought 174
Conclusion 175

Chapter 9 Sprint Retrospective Anti-Patterns 177
Introduction 177
The Scrum Guide on the Sprint Retrospective 178
Sprint Retrospective Anti-Patterns 179
Sprint Retrospective Anti-Patterns of the Scrum Team 179
Sprint Retrospective Anti-Patterns of the Scrum Master 187
Sprint Retrospective Anti-Patterns of the Organization 191
Food for Thought 194
Conclusion 195

Chapter 10 Product Backlog and Refinement Anti-Patterns 197
Introduction 197
The Product Backlog According to the Scrum Guide 198
Common Product Backlog Anti-Patterns 200
General Product Backlog Anti-Patterns 200
Product Backlog Anti-Patterns of the Product Owner 207
Product Backlog Anti-Patterns of the Developers 209
Product Backlog Anti-Patterns of the Scrum Team 211
Food for Thought 212
Conclusion 213

Chapter 11 Sprint Backlog Anti-Patterns 215
Introduction 215
Sprint Backlog Anti-Patterns 216
Sprint Backlog Anti-Patterns of the Scrum Team 217
Sprint Backlog Anti-Patterns of the Developers 220
Sprint Backlog Anti-Patterns of the Product Owner 225
Food for Thought 228
Conclusion 228

Chapter 12 Increment Anti-Patterns 229
Introduction 229
The Purpose of the Increment According to the Scrum Guide 230
Increment Anti-Patterns 231
Increment Anti-Patterns by the Scrum Team 231
Increment Anti-Patterns of the Stakeholders or the Organization 240
Food for Thought 242
Conclusion 242

Chapter 13 Product Goal Anti-Patterns 245
Introduction 245
The Purpose of the Product Goal According to the Scrum Guide 246
Product Goal Anti-Patterns 247
Product Goal Anti-Patterns of the Product Owner 247
Product Goal Anti-Patterns of the Scrum Team 251
Product Goal Anti-Patterns of the Organization 254
Food for Thought 257
Conclusion 257

Chapter 14 Sprint Goal Anti-Patterns 259
Introduction 259
How to Create Sprint Goals 260
Sprint Goal Anti-Patterns 261
Sprint Goal Anti-Patterns of the Scrum Team 262
Sprint Goal Anti-Patterns Induced by the Organization 267

Sprint Goal Anti-Patterns by the Developers 269
Sprint Goal Anti-Patterns by the Product Owner 271
Food for Thought 272
Conclusion 272

Chapter 15 Definition of Done Anti-Patterns 273
Introduction 273
Creating a Successful Definition of Done 275
Definition of Done Anti-Patterns 277
Definition of Done Anti-Patterns of the Developers 278
Definition of Done Anti-Patterns of the Scrum Team 281
Definition of Done Anti-Patterns of the Organization 289
Definition of Done Anti-Patterns of the Product Owner 290
Food for Thought 291
Conclusion 291

Appendix A How to Sabotage Scrum Masters and Product Owners at an Organizational Level 293

Appendix B Toolbox 305

Index 369

Foreword by Dave West

Ken Schwaber, the co-creator of Scrum, often says, "Scrum takes minutes to learn but a lifetime to master." This saying describes the paradox of Scrum: It is a simple framework that is easy to learn and understand, but practicing it well is challenging and takes time. Scrum is not complex, but the environment in which it is employed can be very complex. The complexity that Scrum works within includes the team, the organization, the problem domain, the solution domain, and the stakeholders. Each of these areas can manifest challenges to empiricism, self-management, and the ability to improve. Scrum Teams must overcome these challenges to achieve their full potential using Scrum.

As CEO at Scrum.org, I have seen first-hand the struggles of Scrum Teams. I have seen Product Backlogs that are merely lists of work to be done and Sprint Plans that were dictated to the team instead of being developed by the team to achieve a Sprint Goal. I have seen Product Increments that are intentionally not ready for users and Sprint Reviews in which Stakeholders simply look for progress against a plan instead of understanding the value that was delivered. The list goes on and on. The results in all these situations could have been better for everyone involved if they had understood the common dysfunctions of Scrum Teams and their organization and how to overcome them.

Many organizations look to consultants and practitioners experienced with Scrum to help them avoid these and many other dysfunctions. That's nice if you can find

them; already having made the mistakes and learning from them can help prevent them in the future. In the absence of this personal experience, books, blogs, and articles written by experienced practitioners can help you understand where teams and their organizations can and often do go wrong and what to do about them.

In this book, Stefan describes the anti-patterns he has observed and provides simple and practical advice on their resolution. He has structured the book around the elements of Scrum to make it easy for you, the reader, to easily find anti-patterns. For example, if you are new to the Sprint Review or struggling with it, you can quickly dip into that chapter and see if anything resonates.

Each chapter has many pearls of wisdom, and I have experienced many examples in the organizations I have visited. Still, my favorite two are Scrum Sprint Planning and Sprint Review. Both these events are crucial to the success of Scrum but often need to be better executed, and improvements in these two events have a massive impact on the value of the Increment, team morale, and the relationship with Stakeholders.

In the agile community, we spend much time moaning about the lack of success, blaming the big things. Things include legacy systems, existing processes, HR practices, incentive programs, organization structures, job titles, and team structure. And, yes, many of those things make agile very difficult. But, at the heart of agile is the idea that teams break things down, deliver frequently, and do what they need to do. Agile is much more than teams; but it starts and finishes with teams. How they work, how they collaborate, and the choices they make collectively make them more or less agile. This book is all about those choices. And the first step in any choice is knowing you have a choice to make. A patterns approach is a great way to identify those issues and the choices you have to make.

The phrase “dream big, but start small” is an excellent guide to using Scrum, and I hope this book can provide you with some tips and a frame for changing the world, one choice at a time.

ScrumOn!

— Dave West
October 2023
Weston, MA

Foreword by Janna Bastow

I once found myself amidst the bustling energy of a newly funded startup, right at the awkward juncture of scaling. We had a couple dozen developers who seemed to know their stuff but we weren't jiving as a team yet. Our boss brought in a Scrum Master, complete with a pile of certifications and textbooks and promises of a transformation. And indeed, we did see a transformation! I've never seen a group of four development teams whipped into shape so fast. Their velocity was off the charts… and yet they were delivering absolutely nothing of value. The disconnect between development processes and product needs was massive, so no impact was being made. It was a classic case of a Scrum anti-pattern—mistaking motion for progress.

I wish I'd read this book all those years ago, as Stefan Wolpers' guide would have saved us from a lot of wasted effort.

Over the years, I've worked with hundreds of companies, and I've seen Scrum fail in a spectacular kaleidoscope of ways. Stefan spots all of these pervasive anti-patterns and calls them out, guiding the reader on how to avoid them at every turn. This isn't just another book on Scrum; it's a lifeline for those of us who have found ourselves trapped in a quagmire of "doing Scrum right" but not reaping the benefits. At its heart, this book is all about impact. It's about making use of the tools and

processes that Scrum gives us to get the most out of it for your team, your product, and your business.

But Stefan doesn't stop at the basics. He delves deep, guiding you through the good, the bad, and the downright ugly metrics that will shape your Scrum journey. He tackles tough subjects around stakeholder management with good humor. And for those grappling with the realities of today's distributed teams, there's invaluable advice on establishing Communities of Practice and effective remote management.

One of the standout features that you'll love with this guide is the sheer practicality. It's the kind of book you'll want to keep within arm's reach on your desk, not just for yourself, but to share with your developers, your product owner, your Scrum Master, and anyone else baffled by the Scrum process. Stefan's humorous takes on how to (hypothetically) sabotage your product owner or Scrum Master are not only entertaining but serve as useful cautionary tales, highlighting what not to do (or what to stop doing!). And let's not forget the iconic comics sprinkled throughout the book. These illustrations, both witty and insightful, perfectly capture the essence of the situations
discussed, making your journey to better Scrum more digestible.

In classic Stefan fashion, there's no beating around the bush. Every page, every chapter, is packed with value, devoid of unnecessary fluff. It's a
testament to his expertise and his commitment to delivering actionable insights to the Scrum community.

In essence, if you're looking to make a tangible difference with your work, this book is a must-read. Dive in, apply its lessons, and see the transformation for yourself.

Janna Bastow,
Co-founder & CEO, ProdPad

Preface

The Scrum Anti-Patterns Guide offers a different perspective on how to apply Scrum successfully by applying the principle of "inversion." Instead of explaining how to apply Scrum properly, the book looks at the many things that may go wrong when practicing Scrum—the anti-patterns.

The principle of inversion refers to a change in order or position, specifically, changing perspective to gain different insights. This principle is a powerful learning tool because it encourages viewing problems from an alternative standpoint, revealing new dimensions and possible solutions.

In the ideal case, inversion helps us confront the constraints, potential pitfalls, or worst-case scenarios upfront, which can illuminate the path to success by highlighting what to avoid.

Even under less-than-ideal circumstances, the principle of inversion can still provide immense value, even when you're already making the mistakes you should avoid. Here's how:

Identifying root causes: Through inversion, you can better understand the root causes of the mistakes. Often, we focus on symptoms rather than causes. By asking,

"What actions or conditions led to these mistakes?" you can start to address the underlying issues.

Preventing recurrence: Once the mistakes are made, inversion can help prevent them from recurring. By analyzing what led to the error, you can implement safeguards and checkpoints to prevent the same mistakes in the future.

Learning and adaptation: Making mistakes is a part of the learning process. Inversion promotes learning from these mistakes and adapting your practices accordingly. The goal is not to avoid all mistakes but to continuously learn and improve.

Prioritization of actions: Not all mistakes have the same impact. Inversion can help identify the most damaging errors, effectively prioritizing countermeasures.

Promoting a proactive mindset: Rather than merely reacting to errors, inversion encourages a proactive mindset, fostering preventative actions and forward-thinking solutions.

Consequently, applying the inversion principle to Scrum can produce profound results as Scrum is grounded in feedback-informed learning and continuous improvement, concepts that align well with inversion.

The book started as a series of blog posts in 2017/2018 when I began curating all my observations over the years, particularly in larger organizations.

Register your copy of ***The Scrum Anti-Patterns Guide*** on the InformIT site for convenient access to updates and/or corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account. Enter the product ISBN (**9780137977963**) and click Submit. Look on the Registered Products tab for an Access Bonus Content link next to this product, and follow that link to access any available bonus materials. If you would like to be notified of exclusive offers on new editions and updates, please check the box to receive email from us.

Acknowledgments

I want to thank Scrum.org for supporting this book, particularly David West and Kurt Bittner.

Another big thank you goes to Haze Humbert of Pearson for her inexhaustible patience.

Finally, I would like to thank Oliver Guillard for turning my sketches into useful graphics.

Join the Community

If you would like to talk to other readers of this book, consider joining the community. All you have to do is provide your credentials, and I will invite you.

You can join via this link: https://bit.ly/sag-community. Alternatively, use the following QR Code, see Figure 00.1.

Figure 00.1 Sign up for the Reader Community.

About the Author

Stefan Wolpers is a Professional Scrum Trainer at Scrum.org, Agile Coach, and Scrum Master, specializing in guiding agile transformations through practices like Scrum, LeSS, Kanban, Lean Startup, and professional product management. He's a licensed Agile Fluency™ Team Diagnostic facilitator with a history of senior leadership roles.

His agile coaching focuses on scaling product delivery for fast-growing, venture-capital-backed startups as well as transitioning existing teams in established enterprises.

Stefan curates the widely followed "Food for Agile Thought" newsletter, engaging 50,000+ global Agile enthusiasts. He shares insights on Age-of-Product.com, hosts a vibrant Slack community of 18,000+ agile practitioners, and has authored e-books on agile themes, garnering over 100,000 downloads.

This page intentionally left blank

Introduction

Purpose of This Book

The primary purpose of *The Scrum Anti-Patterns Guide: Challenges Every Scrum Team Faces and How to Overcome Them* is to help Scrum practitioners identify, understand, and address commonly observed patterns of behavior that undermine the effectiveness of Scrum in achieving its goals.

The book addresses these challenges—from the misaligned understanding of roles and practices to organizational issues—highlighting these anti-patterns and pointing to practical solutions. The guide seeks to enhance the application of Scrum, making it more effective in driving innovation, productivity, and overall value delivery.

In real life, stakeholders are not usually interested in how we solve their problems as long as we ethically play by the rules of our society in alignment with our organization's culture. Instead, they are interested in the regular delivery of valuable Increments.

In other words, we do not get paid to practice Scrum but to solve our customers' problems within the given constraints while contributing to the sustainability of our organization. Unfortunately, this typical attitude also implies that no matter how

well versed your Scrum Team is in practice, no one outside your team will care unless your team regularly delivers valuable Increments in every Sprint.

So, Scrum can contribute to solving your customers' problems if you choose Scrum in the proper context: solving complex adaptive problems. Notably, Scrum can also be a complete waste of everyone's time when selected for the wrong reasons: If you know exactly what to build, there is no need to employ Scrum.[1]

As if choosing the proper context for employing Scrum wasn't challenging enough, Jeff Sutherland and Ken Schwaber designed Scrum's framework as "purposefully incomplete."[2] While this is true of any framework—otherwise we would call them methodologies—many practitioners interpret the phrase as an invitation to change and augment the framework in basically any way they see fit, delivering value being the justifying means; see Figure INT.1.

Figure INT.1 A "purposefully incomplete" framework will result in many different approaches, opinions, and practices on complementing Scrum best, often contradicting each other or disrespecting first principles.

1. Scrum excels at figuring out what is worth building while mitigating risk. Consequently, its iterative and incremental approach comes at a price; we plan a lot in Scrum, which is not free.
2. Ken Schwaber and Jeff Sutherland, "Scrum," in The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#scrum-definition.

The Scrum Anti-Patterns Guide seeks to equip all practitioners with the insights and strategies necessary to transform these anti-patterns into learning, growth, and continuous improvement opportunities. The book comprises 15 chapters and two appendices:

Chapter 1: Scrum Master Anti-Patterns studies points such as the "agile" manager, tracking flawed or useless metrics, and defining technical solutions to problems likely stemming from misunderstandings, dogmatism, impatience, indifference, opportunism, frustration, or inexperienced practice.

Chapter 2: Product Owner Anti-Patterns dissects observations such as monopolizing Product Backlog management, poor communication with stakeholders and teammates, and overemphasizing administrative tasks, issues that critically limit the Product Owner's potential to maximize the product's value.

Chapter 3: Scrum Developer Anti-Patterns reminds us of Developers' crucial contribution to making a Scrum Team successful. Nevertheless, anti-patterns such as corner-cutting to meet deadlines, side gigs, and delivering undone work undermine Developers' effectiveness.

Chapter 4: Scrum Stakeholder Anti-Patterns points to the significant influence that problems at this level, such as a lack of transparency, inadequate leadership support, and cherry-picking individual practices, have on Scrum Teams, inhibiting the team's progress toward its Product Goal.

Chapter 5: Sprint Anti-Patterns addresses challenges to providing a framework to deliver value and foster predictability. Anti-patterns such as variable Sprint length, undermining a Scrum Team's autonomy, and bypassing the Product Owner negatively affect the result.

Chapter 6: Sprint Planning Anti-Patterns delves into familiar problems such as the lack of capacity checks, output focus, and imposed forecasts, which hinder Scrum Teams from aligning on delivering customer value.

Chapter 7: Daily Scrum Anti-Patterns reminds us of the event's purpose: facilitating inspection, adaptation, and short-term planning toward the Sprint Goal. Anti-patterns such as treating it as a status report, assigning tasks directly to Developers, and stakeholder interference can inhibit the Daily Scrum's effectiveness.

Chapter 8: Sprint Review Anti-Patterns stresses the importance of the Scrum event for assessing progress toward the Product Goal. Unfortunately, anti-patterns such as customers' absence, approval gates, and passive stakeholders can undermine this approach.

Chapter 9: Sprint Retrospective Anti-Patterns examines the crucial event for continuous Scrum Team improvement. Yet, anti-patterns such as rushed Retrospectives, unsuitable venues, and the absence of psychological safety can render its outcome questionable at best.

Chapter 10: Product Backlog and Refinement Anti-Patterns analyzes typical issues of many Scrum Teams, from an oversized Product Backlog, including outdated items, to excessive detailing, to a lack of regular refinement, impeding the core artifact crucial for Scrum Team success.

Chapter 11: Sprint Backlog Anti-Patterns points to the Sprint Backlog's temporary nature as an actionable repository and plan for achieving the Sprint Goal. Consequently, anti-patterns such as overly detailed planning, fixed scope, and changing work items mid-Sprint can undermine its effectiveness.

Chapter 12: Increment Anti-Patterns looks at Scrum's advocating for regular releases of Increments for swift user feedback to accelerate learning. Anti-patterns such as postponed releases, releasing exclusively to the complete user base, and Increments not meeting the Definition of Done can easily disrupt this crucial iterative process.

Chapter 13: Product Goal Anti-Patterns delves into the Product Goal's purpose of aligning strategy to tasks and promoting focus and transparency. Yet, anti-patterns, from imposed Product Goals to the lack of communication, can undermine these intentions.

Chapter 14: Sprint Goal Anti-Patterns analyzes how the Scrum Team is impeded from delivering outcomes valued by customers by unsuited Sprint Goals, overambitious plans, inconsistent achievements, and misplaced focus.

Chapter 15: Definition of Done Anti-Patterns underscores the Definition of Done's crucial role in Scrum, as it helps align the team on product quality standards, fosters continuous improvement, and creates value. Its absence or inadequacy, however, may expedite failure.

Appendix A: How to Sabotage Scrum Masters and Product Owners at an Organizational Level uses a reverse-thinking exercise to identify and tackle

potential challenges Scrum Masters and Product Owners might face in organizations.

Appendix B: Toolbox provides many useful tools for practitioners in Scrum, including interview questions to help newly appointed Scrum Masters get up to speed; ways to build a community of practice, create a Definition of Done, apply Liberating Structures to remote Scrum events; and more.

You can read the book cover to cover or refer to an individual chapter; its content is "snackable."

Who Should Read This Book

The Scrum Anti-Patterns Guide is designed as an essential resource for

- Scrum Masters seeking to deepen their practice,
- Product Owners aiming for a valuable Product Goal, an actionable Product Backlog, and improved stakeholder relations, and
- Developers striving to create more customer value.

By providing a deep understanding of potential pitfalls and offering practical solutions to overcome them, this book empowers all Scrum practitioners to maximize the value of Scrum and foster a culture of continuous improvement.

How This Book Is Organized

The book is structured as a repository of 180-plus insights based on Scrum roles, events, artifacts, and commitments in the form of snackable content. While you can read it sequentially from beginning to end, it is not written to be consumed in that way.

When you want to better understand what challenging behaviors and practices you may encounter in a particular setting—for example, if you are having trouble with your Sprint Review—refer to the respective chapter. Use the chapter as a briefing on

the possible reasons for teammates', stakeholders', and managers' behavior and how those anti-patterns may be overcome. Typically, you will find more information on the topic in numerous footnotes pointing to additional details.

You may find reading *The Scrum Guide Reordered*[3] alongside this book useful to compare theoretical patterns derived from Ken Schwaber and Jeff Sutherland *Scrum Guide*—how Scrum is supposed to be—to the reality in your organization. *The Scrum Guide Reordered* is based on about 95 percent of the text of The 2020 Scrum Guide, extended with categories such as self-management, commitments, and accountability.

Besides the Scrum anti-patterns in this book, Appendix B, "Toolbox," contains a set of practices and exercises that are well-suited to address some of the underlying reasons for the described Scrum anti-patterns.

The map of Scrum anti-patterns presented in this book is not the "territory" but merely an abstraction of reality[4] to help you learn and continue the exploration. Every organization applying Scrum faces its unique set of challenges. There is no one-size-fits-all approach to solving them.

3. Stefan Wolpers, *The Scrum Guide Reordered,* 2020, https://age-of-product.com/scrum-guide-reordered.
4. Shane Parrish and Farnam Street, "The Map Is Not the Territory," https://fs.blog/map-and-territory.

1 Scrum Master Anti-Patterns

Introduction

The reasons Scrum Masters violate the spirit of the Scrum Guide are multifaceted. Typical Scrum Master anti-patterns run from ill-suited personal traits to complacency, pursuit of individual agendas, and misunderstanding of the role itself; see Figure 1.1.

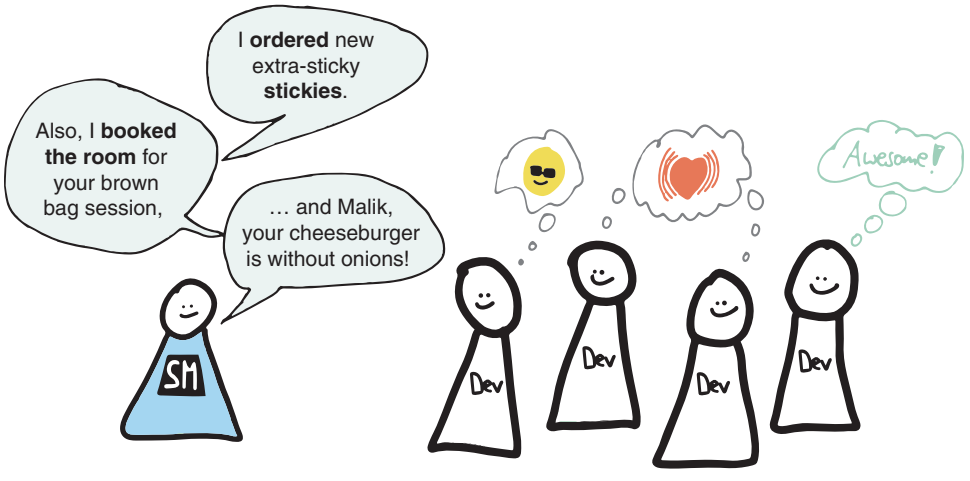

Figure 1.1 Beware of becoming a Scrum parent.

THE SCRUM MASTER ACCORDING TO THE SCRUM GUIDE

Before we start dissecting probable reasons for and manifestations of Scrum Master anti-patterns, let us revisit how the Scrum Guide defines the role of the Scrum Master:[1]

- Scrum Masters are accountable for establishing Scrum, aiding all in understanding its theory and practice.
- They are responsible for the Scrum Team's effectiveness, enabling practice improvements within the Scrum framework.
- Scrum Masters act as servant leaders for the Scrum Team and the larger organization.
- They serve the Scrum Team by coaching self-management and cross-functionality, focusing on high-value Increments, removing impediments, and ensuring effective Scrum events.
- They aid the Product Owner through effective Product Goal definition and Product Backlog management and by promoting understanding of clear backlog items, enabling empirical planning, and facilitating stakeholder collaboration.
- Serving the organization, Scrum Masters lead Scrum adoption, plan and advise on implementations, promote understanding of empirical approaches, and remove barriers between stakeholders and Scrum Teams.

The keystone of the Scrum Master role is leadership. (Too bad that it is officially no longer called "servant leadership"; I found that description to be better suited.) Unfortunately, in many cases of Scrum Master anti-patterns, it is precisely this idea that an individual does not meet.[2]

POSSIBLE REASONS WHY SCRUM MASTERS LEAVE THE PATH

Following are some often-observed reasons for Scrum Masters not behaving as intended:

- **Ignorance or laziness:** Some Scrum Masters assume that one approach to Scrum fits every team. They learned Scrum in a specific context and apply that same

1. Ken Schwaber and Jeff Sutherland, "Scrum Master," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#scrum-master.
2. See also the detailed lists of services rendered to the Product Owner, the Developers, and the organization by the Scrum Master, as defined by the Scrum Guide.

approach in every organization in which they are active, no matter the circumstances. Why go through the hassle of teaching, coaching, and mentoring if you can shoehorn the "right way" directly into any Scrum Team?

- **Lack of patience:** Patience is a critical resource that a successful Scrum Master needs to exhibit in abundance. But, of course, there is no fun in readdressing the same issue several times, rephrasing it probably, if the solution is obvious—from the Scrum Master's perspective. So, why not tell the team how to do it "right" all the time and save all that effort? Too bad that Scrum cannot be pushed but needs to be pulled—that's the essence of self-management. (You may observe a similar behavior resulting from boredom.)
- **Dogmatism:** Some Scrum Masters believe in applying the Scrum Guide literally, which unavoidably causes friction because Scrum is a framework, not a methodology; see Figure 1.2. Nevertheless, teaching Scrum that way feels good: team members come and ask for help; now, the Scrum Master has a purpose:
  - When Scrum Team members follow the rules, the Scrum Master has influence or authority.
  - Being among the chosen few who interpret the Scrum Guide "correctly" secures status and respect among teammates and the broader organization.
  - The Scrum Master may easily attribute the Scrum Team's progress or success to the Scrum Master's teaching; now, they also have proof regarding their mastery of Scrum.
  - Finally, their mastery of Scrum is a convincing argument for the organization to keep the dogmatic Scrum Master on the payroll; apparently, the Scrum Teams need an interpreter of the Scrum Guide to reap the framework's benefits.
- **Laissez-faire turned into indifference:** Pointing the team in a direction where the team members can find a solution for an issue is good leadership. However, letting them run without guidance all the time sooner or later may turn into indifference or an I-do-not-care mentality.
- **The opportunist:** Secretly, the Scrum Master believes that this Scrum thingy is a fad but recognizes that it offers a well-paid position: "I will weather the decline in demand for project managers by getting a Scrum Master certificate. How hard can this be—the manual is merely 13 pages?" However, this conviction will inevitably bring out the opportunist's true colors over time.

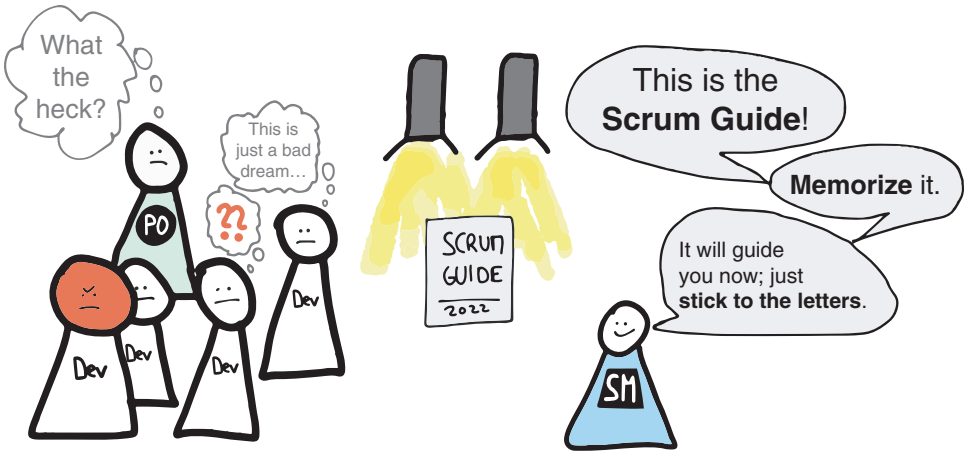

Figure 1.2 Scrum is a practical yet imperfect tool to solve complex, adaptive problems but is not an infallible methodology.

- **Frustration:** The Scrum Master has been working diligently for months, but the team is not responding to the effort. The level of frustration is growing; see Figure 1.3. There are many potential reasons for a failure at this level. Here are just a few:
  - The agile effort lacks sponsorship from the C-level of the organization.
  - There is a widespread belief that Agile is just the latest management fad and thus is ignorable.
  - The team composition may be wrong—perhaps some team members despise Scrum.
  - There is no psychological safety to address the team's problems.
  - The company culture is antagonistic toward transparency and radical candor.
  - Individual team members harbor personal agendas unaligned with the team's objective.

  Ultimately, the Scrum Masters cannot solve this issue by themselves; its fixing is an effort of the whole Scrum Team.
- **The tactical Scrum Master:** These Scrum Masters drank HR's Kool-Aid and believe that Scrum Master is a position, not a role. Moreover, there is a career path from junior Scrum Master to VP of Agile Coaching. Consequently, they constrain their work strictly to the Scrum Team level until being promoted.

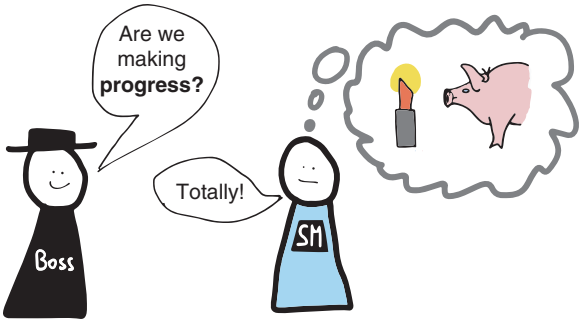

Figure 1.3 Scrum Masters are only human, too. Lend them your hand when things make no progress.

- **The rookie:** The Scrum Master might merely be inexperienced. Given that we all need to learn new skills regularly, cut them some slack in this case, and reach out to support their learning effort. Scrum is a journey, not a destination, and no one on the team travels alone.

ANTI-PATTERNS FROM ACTING AS AN AGILE (LINE) MANAGER

Agile Management is an oxymoron: you manage systems but lead people. The primary purpose of any agile practice is to empower those closest to a problem to find a solution and to help the team to self-manage over time. Self-organizing teams need coaches, mentors, and servant leaders; they do not need directive managers. Watch out for the Scrum Master anti-patterns corresponding to this agile manager attitude.

1. Agile Manager

Observation: Your Scrum Master acts like an agile line manager, dismissing Scrum's idea of self-managing teams of capable individuals who solve customer problems autonomously.

Background: Self-management does not mean the absence of management: Why would a Scrum Master or Scrum Team assume, for example, responsibility for payroll? Would that help with creating value for the customer? No. Being a self-managing team does not mean the absence of management per se, it simply means a different kind of management that is focused on helping people to grow their skills rather than telling people how to do their work.

There are various reasons why a Scrum Master may behave in this way; for example:

- **Lack of understanding:** Misunderstanding the Scrum Guide can lead a Scrum Master to act as a line manager. They may not fully comprehend the principle of self-managing teams due to a lack of training, coaching, and mentoring.
- **Fear of loss of control:** Some Scrum Masters, due to their previous roles as managers, may fear losing control and relevance. They need to learn and embrace the idea that by helping the team to self-manage, their ability to have a positive effect is multiplied, not reduced.
- **Organizational culture:** Scrum Masters may resort to old managerial behaviors if the organization still values and rewards traditional management practices. This behavior may persist, especially if the organization hasn't fully embraced agile principles, particularly building projects and products around motivated individuals and trusting them to do the job.
- **Resistance to change:** Change can be uncomfortable and difficult. Some "Scrum Masters" may resist the change to a servant leadership role, as stated in the Scrum Guide, from a traditional command-and-control management style due to ingrained habits, comfort, and familiarity.
- **Lack of trust:** A Scrum Master who doesn't trust the team's ability to self-manage might act as a line manager. This behavior contradicts the Scrum Guide's vision of a Scrum Master serving the team and fostering an environment where the team can work at its highest potential.
- **Overemphasis on delivery:** If the organization focuses overly on meeting deadlines and delivery, the Scrum Master may adopt a managerial approach to push the team, contrary to promoting sustainable development.
- **Insufficient support:** Without enough support from upper management and other departments to foster an agile environment, a Scrum Master may revert to line management out of frustration or necessity. This scenario contradicts the Agile Manifesto's principle of giving the team the needed environment and support.

Remedy: Some steps the organization and the teammates can take to help the Scrum Master overcome the "agile manager" behavior are, for example:

- **Mentorship and coaching:** Pair the Scrum Master with an experienced mentor to help them learn and adopt appropriate behaviors. The mentor can give real-time

feedback, helping the Scrum Master to understand when they're slipping into a line management role.

- **Continuous learning and training:** Promote ongoing education to ensure the Scrum Master understands the principles of the Scrum Guide and the Agile Manifesto. They need to comprehend that the role of the Scrum Master is to serve the team, not manage it. Regular refresher courses and workshops can reinforce these principles.
- **Encourage dialogue:** Team members can initiate a conversation with the Scrum Master about their concerns, being open and respectful. It's important to refer to the Agile Manifesto and Scrum principles when discussing how the Scrum Master's behavior affects the team's self-management.
- **Feedback:** Consider implementing a 360-degree feedback system, allowing team members to provide anonymous feedback to the Scrum Master if team members feel uncomfortable addressing the issue in Retrospectives.
- **Performance metrics:** Align Scrum Master performance metrics with agile principles. Instead of assessing based on output and meeting delivery deadlines, focus on team satisfaction, quality of work, and progress toward self-management and accomplishing the team's Product Goal.
- **Patience and persistence:** It takes time to unlearn ingrained behaviors and adopt new ones. Be patient with Scrum Masters who are transitioning from a traditional management role. Regularly reinforce the importance and benefits of agile practices while addressing any concerns or resistance they might have.

> ***Food for Thought:*** *As a Scrum Master, gaining work experience in a Scrum Team as a Developer or Product Owner might be helpful in building empathy.*

2. Running Scrum Events by Allowing Someone to Speak

Observation: The Scrum Master hosts all Scrum events down to the level where team members wait for permission to contribute.

Background: When team members seek approval from the Scrum Master before speaking out, the Scrum Master has already left the facilitation role in favor of the supervisor mode. While Scrum Masters support their teams by facilitating events, it does not imply that they always run the show with a domineering attitude.

Their job is to help their teammates to accomplish the purpose of the respective Scrum event, not to enforce an orderly procedure from their perspective.

Remedy: There are a few things the teammates can do to encourage their Scrum Master to relinquish an overly controlling role:

- **Initiate open discussion:** Address the issue respectfully within the team, perhaps during a Retrospective. Make the conversation about the process and roles, not the person.
- **Assert autonomy:** Begin contributing without waiting for the Scrum Master's prompt. Show initiative in discussing and deciding tasks, thus reinforcing the concept of a self-managing team.
- **Encourage peer facilitation:** Suggest a rotation of the facilitator role for Scrum events to promote shared ownership and demonstrate that the Scrum Master doesn't always have to be in control.
- **Educate on role:** Gently remind the Scrum Master about the core aspects of their role: serving the team and the Product Owner, removing impediments, and not managing tasks or people.
- **Seek external guidance:** If internal efforts aren't effective, consider contacting an agile coach or experienced Scrum Master from another team for advice or mediation.

Each of these steps aligns with the values of the Agile Manifesto, encouraging individuals and interactions over processes and tools and promoting a more effective, collaborative way of working.

3. The Administrative Assistant

Observation: The Scrum Master pampers the Scrum Team to a level that keeps the team dependent on their services. For example, they might

- Organize meetings,
- Purchase stickies and Sharpies,
- Take notes, write protocols,
- Update the Product Backlog on behalf of the team (whether in a tool or not),
- Organize budget for tools, education, and team activities,

- Preselect applicants for open team positions, and
- Channel stakeholder communication—you get the idea.

Background: Sometimes, Scrum Masters believe that being supportive of their teams requires assuming the role of the personal team assistant. That is an unfortunate misinterpretation of their role; they are supposed to help their teammates become self-managing, and everyone knows how to order or "organize" stickies and Sharpies. More critical, however, is when Scrum Masters decide to keep the team in the dark about principles and practices in order to secure their job.

What might be the motivation for such behavior? Some aspects are as follows:

- **Desire for power and control:** The Scrum Master may relish the feeling of indispensability and control they gain by keeping the team reliant on their services, appearing more knowledgeable and indispensable.
- **Positioning:** If they view administrative tasks as "beneath" the team members, they might attempt to create an image of themselves as self-sacrificing leaders by taking on these tasks.
- **Avoidance of accountability:** By focusing on administrative tasks and not their actual responsibilities—such as coaching the team and the wider organization about Scrum—they might be sidestepping more challenging, accountability-laden aspects of their role.
- **Comfort in familiar tasks:** They might find comfort in administrative tasks, especially if they previously held roles that required such tasks, giving them a sense of control and accomplishment.
- **Lack of confidence:** They might lack confidence in the team's ability to manage these tasks or coach and facilitate effectively, causing them to retain control over seemingly trivial matters.

Remedy: Similar to the Agile Manager anti-pattern, there are a few steps the teammates can take to help the Scrum Master overcome this behavior:

- **Peer feedback:** Offer constructive feedback to the Scrum Master. Encourage teammates to do the same, creating a unified voice that calls attention to the issue.
- **Proactive participation:** Don't wait for the Scrum Master to assume tasks. Proactively take ownership of tasks such as updating the task board or organizing events, demonstrating your ability to self-manage.

- **Ask for coaching:** Request that the Scrum Master focus more on facilitating, coaching, and mentoring than on taking over administrative tasks. Suggest working sessions to help deepen understanding of Scrum principles and practices.
- **Education:** Share resources such as books, articles, or videos on the role of a Scrum Master, or organize team sessions with Scrum Masters from other teams where they share their success patterns.
- **Engage management:** If the Scrum Master's behavior persists, you may need to involve senior management or human talent department. Maintaining a productive work environment is crucial, and sometimes it requires outside intervention to resolve such issues.

The aim is not to antagonize the Scrum Master but to help them understand and fulfill their role more effectively. Always approach these steps with respect and an eye toward improving team dynamics and productivity.

4. Tracking Flawed or Useless Metrics

Observation: The Scrum Master tracks metrics that focus on output or individual performance, like the number of story points completed, personal velocity, or estimation accuracy.

Background: Generally speaking, metrics help to clarify the current situation and allow us to gain insight on change over time. Without metrics, assessing any effort or development will be open to gut feeling and bias-based interpretation.

A metric should therefore be a leading indicator for a pattern change, providing an opportunity to analyze the cause in time. The following three general rules for agile metrics have proven to be useful:

1. Track only those metrics that apply to the team. Ignore those that measure the individual.
2. Do not measure parameters because they are easy to track. (Various project management tools offer out-of-the-box reports.)
3. Record context as well. Data without context, for example, the number of available team members or the intensity of incidents during a Sprint, may turn out to be nothing more than noise.

While utilizing predefined reports of the team's Product Backlog management software is probably a sign of ignorance or laziness, keeping track of individual performance metrics—such as story points per Developer per Sprint and reporting them to that person's line manager—is a critical warning sign.

Remedy: What action may a Scrum Team take to convince the Scrum Master to switch to measuring meaningful agile team metrics?

There are a few options; for example:

- **Explain metrics in context:** The team can educate the Scrum Master about agile values, principles, and practices and their impact on identifying agile metrics. For example, it's essential to highlight that individual-focused metrics undermine team collaboration and often lead to poor results.
- **Propose relevant metrics:** The team can propose tracking more relevant and beneficial metrics such as lead time, cycle time, or team health. These metrics focus more on the team's performance and value delivery than on individual output. It is helpful if the team also explains why these metrics are more meaningful and how they improve the team's effectiveness and efficiency.
- **Self-organization:** As a self-managing team, members can take control of the metrics themselves. They can start tracking the proposed metrics and demonstrate their value over several Sprints. Seeing the positive results might convince the Scrum Master to change their approach.
- **External facilitator:** If the situation does not improve, the team could suggest having an external agile coach or another experienced Scrum Master facilitate a workshop. The team and Scrum Master can openly discuss the issue and find a solution. The external facilitator may be able to provide a neutral viewpoint and effective advice.

5. Spying for Management

Observation: During the Sprint, the Scrum Master reports to the management whether the Scrum Team will meet the current forecast.

Background: I took this from a job offer I received: "You will coordinate and manage the work of other team members, ensuring that timescales are met and breaches are escalated."

Besides completely misunderstanding the Scrum Master's role, there are some other reasons that may motivate a Scrum Master to disregard Scrum values:

- **Pressure from management:** The Scrum Master may be under pressure from management to provide frequent updates and assure progress. This demand could stem from a lack of understanding of Agile and Scrum principles among the leadership, particularly the emphasis on trust and empowerment over monitoring and control.
- **Lack of trust in team:** The Scrum Master may lack trust in the team's ability to meet their forecast and thus feels the need to keep management informed. The Scrum Master's lack of trust may even contribute to their ineffectiveness by reducing their motivation.
- **Insecurity about role:** The Scrum Master might feel insecure about their role and value within the organization. By reporting to the management, they may try to demonstrate their importance and control over the team's work.

Remedy: What action should a Scrum Team take to convince the Scrum Master that this behavior is inappropriate? I would start with the first and second of these suggestions and keep the third in reserve:

1. **Communicate openly:** Team members should initiate a general conversation with the Scrum Master about their concerns, explaining how this behavior undermines trust and self-management principles. They should pursue this respectfully and constructively, using specific instances as examples, for instance, during a Retrospective. The Retrospective's outcomes could also be shared with the broader organization, helping others understand the team's way of working and their commitment to the Scrum values.
2. **Self-reporting:** The team can propose a self-reporting mechanism whereby they take responsibility for reporting their progress. The process may include visual radiators or dashboards that everyone, including management, can access. In addition, the team may consider publishing a newsletter to make stakeholder communication as simple as possible.
3. **Involve a neutral party:** If direct communication doesn't yield results, involving a neutral party like an agile coach or a trusted leader in the organization could be helpful. They can facilitate a discussion and help resolve the issue.

6. Team Harmony over Conflict Resolution

Observation: The Scrum Master sweeps conflict and problems under the rug by not using Sprint Retrospectives to address them openly.

Background: There are several reasons why a Scrum Master reverts to peacekeeping; for example:

- **Overemphasis on positivity:** The Scrum Master may erroneously believe that Retrospectives should focus only on what went well to keep the team's morale high. However, this view neglects the equally important aspect of addressing what didn't work and finding ways to improve, a key element of the Agile Manifesto's emphasis on continuous improvement.
- **Fear of confrontation:** The Scrum Master may be uncomfortable with conflict and avoid addressing it openly. This behavior often arises from a misunderstanding of the nature of conflict, viewing it as harmful rather than an opportunity for improvement and team growth.
- **Preservation of the status quo:** The Scrum Master might be afraid of upsetting the existing emotional balance of the team, even if this equilibrium isn't producing the best results. This idea may result from a lack of understanding of the continuous improvement principles underpinning Agile and Scrum.
- **Personal relationships:** If the Scrum Master has close personal relationships within the team, they may fear damaging the relationship by bringing up contentious issues, directly violating the Scrum value of openness.
- **Misunderstanding the role:** The Scrum Master may not fully understand their role and responsibility to facilitate effective Retrospectives. They may see their role as a peacekeeper rather than an improvement facilitator.
- **Lack of skills or training:** The Scrum Master may lack the necessary skills to facilitate a Retrospective effectively, such as being able to manage conflicts, drive constructive discussions, and aid the team in deriving actionable insights. This skill gap points to a need for further professional development in these areas.
- **Fear of repercussions:** The Scrum Master may be apprehensive about the potential fallout from exposing issues. This fear could be due to a corporate culture that penalizes mistakes rather than viewing them as learning opportunities, contradicting Agile's principle of regular reflection for effective adjustment.

- **Avoidance of accountability:** By not discussing issues openly, the Scrum Master may be trying to avoid assigning or taking accountability for problems, which directly conflicts with the Scrum value of commitment to achieving team goals.
- **Belief in privacy:** The Scrum Master may believe that certain issues are private or sensitive and thus not suitable for open discussion in a Retrospective.

Remedy: So, what can we do about this anti-pattern? How can we help the Scrum Master overcome their flawed understanding of Scrum values, openness, conflict, and Retrospectives? A few things come to mind; for example:

- **Feedback and communication:** Encourage the team to openly communicate with the Scrum Master about how their action of ignoring conflicts is affecting the team's performance and morale. Honest feedback may serve as a wake-up call.
- **Coach for transparency and openness:** Foster an environment where transparency, openness, and constructive feedback are encouraged and appreciated. Show the Scrum Master the value of these practices by modeling them in action.
- **Promote a learning culture:** Cultivate a culture that views mistakes and conflicts as opportunities for learning and growth, not as failures. This approach aligns with the Scrum principle of inspecting and adapting to improve.
- **Address fear of conflict:** If the Scrum Master avoids conflict, address this issue directly. Discuss the benefits of constructive conflict and how it leads to better solutions and stronger team cohesion.
- **Invite an agile coach or experienced Scrum Master:** In persistent cases, it could be beneficial to invite an experienced agile coach or Scrum Master to guide your Scrum Master. They can provide an outside perspective and suggest effective ways to handle conflicts and problems.

At the same time, be also aware that self-management is a privilege earned through delivering valuable outcomes. Suppose the team fails to deliver due to internal strife and proves incapable of overcoming the situation. In that case, there may be a moment when a Scrum Team should take action against an ineffective Scrum Master, Product Owner, or any team member.

If a team can't improve or fix itself, stakeholders may lose confidence, cancel the project, and potentially dissolve the team, emphasizing the responsibility of management to intervene when necessary.

Scrum Master Anti-Patterns by Scrum Events

The Sprint Planning

The anti-patterns in this section focus on the Sprint Planning.

7. No Slack Time

Observation: The Developers regularly bow to the hard-pushing Product Owner: "Last Sprint, you delivered twenty-five story points, and now you are choosing only seventeen?" Consequently, they accept more issues into the Sprint Backlog than they can stomach without the Scrum Master's intervention.

See Chapter 3, anti-pattern 8, for a detailed description.

8. Unrefined Work Items

Observation: The Scrum Master does not address accepting "unrefined" Product Backlog issues into the Sprint Backlog during Sprint Planning. (This observation refers to regular work, not emergencies or last-minute changes of high importance.)

Background: The Scrum Master's choice increases the risk that the Developers will probably not accomplish the Sprint Goal, rendering a core Scrum principle useless: reliably providing valuable, potentially shippable Increments every single Sprint.

What reasons may a Scrum Master have to ignore the issue? Some things come to mind:

- **Overreliance on team autonomy:** The Scrum Master may think it is the Developers' sole responsibility to identify the Product Backlog items for the Sprint Backlog without interference.
- **Complacency with current practices:** The Scrum Master might be complacent with the current practices, especially if the team has delivered despite the absence of refined Product Backlog items. However, this practice goes against the Agile Manifesto's principle of continuous attention to technical excellence and good design.
- **Perceived lack of time:** The Scrum Master may believe there isn't enough time for thorough refinement, not objecting to the Developers' pushing ahead with unrefined Product Backlog items.

- **Lack of experience:** A Scrum Master new to the role may not fully understand the implications of accepting unrefined Product Backlog items into the Sprint Backlog, underestimating the associated risk.
- **Avoidance of conflict:** The Scrum Master might avoid conflict within the team or with stakeholders by not insisting on refinement. They may perceive that challenging the Developers' decision may lead to disagreements.

Remedy: The inherent risk that repeating an exception turns it into the new normal is obvious. So, how can we counter this anti-pattern of not addressing the increased risk of accepting unrefined Product Backlog items into the Sprint?

Some suggestions are as follows:

- **Promote the benefits of refinement:** The Scrum Team can work together to emphasize the value of Product Backlog refinement. Highlighting its role in reducing uncertainties, managing risks, and enabling more accurate estimations may help the Scrum Master understand its importance.
- **Establish refinement practices:** As a Scrum Team, schedule regular Product Backlog refinement sessions to ensure this activity becomes a part of the team's routine. The Scrum Master can facilitate these sessions until the team can handle them autonomously.
- **Foster transparency:** Continually reinforce the Scrum value of transparency. Show how it's essential for every team member to understand what the team will work on and the potential risks associated with unrefined Product Backlog items.
- **Promote shared understanding:** Ensure that everyone on the Scrum Team has a shared understanding of each Product Backlog item before it's pulled into a Sprint. The Scrum Team can support the creation of a shared understanding by having discussions, asking questions, and seeking clarifications during Sprint Planning and Product Backlog refinement sessions.
- **Feedback and inspection:** Encourage regular inspection of the process and feedback. If the team experiences negative consequences from working on unrefined Product Backlog items, use this opportunity to discuss the issue and brainstorm solutions.
- **Encourage stakeholder education:** If external pressure is an issue, consider sessions where the Scrum Team educates stakeholders about the principles of Scrum and the importance of Product Backlog refinement. Make it clear that pushing unrefined

items into the Sprint can hurt the quality of the product, the predictability of the team's work, and can hinder accomplishing the Sprint Goal in general.

- **Consider a Definition of Ready:** Finally, the team could agree on a Definition of Ready, specifying the conditions a Product Backlog item must meet before Developers can select it during Sprint Planning. (However, beware of turning such a practice into a dogmatically applied approval gate.) Learn more about the Definition of Ready in Chapter 15, anti-pattern 3.

9. No Improvements

Observation: While the Scrum Team agrees on improvement items during Retrospectives, they never consider them during Sprint Planning.

See Chapter 6, anti-pattern 16, for a detailed description of this anti-pattern.

THE SPRINT

The following anti-patterns focus on the mishandling of the Sprint itself.

10. Flow Disruption

Observation: The Scrum Master is unable to prevent stakeholders from disrupting the workflow of the Scrum Team during the Sprint.

Background: There are many manifestations of how stakeholders can interrupt the Scrum Team's flow during a Sprint, impeding the team's productivity and endangering accomplishment of the Sprint Goal. Some classic examples follow:

- **Misusing Scrum events:** Stakeholders or managers turn the Daily Scrum into a reporting session.
- **Disrupting Developers:** Stakeholders constantly address Developers directly during the Sprint, ignoring that interruptions harm the Scrum Team's effectiveness.
- **Team membership disruption:** Line managers take team members off the Scrum Team, assigning them to other tasks, or add members to the Scrum Team without prior consultation of the team members.

The last example is particularly challenging, as creating a Scrum Team is expensive due to the inevitable drop in productivity during the norming and storming phases of the team-building phase. Hence, changing their composition is a critical decision

that shall include the team members. Scrum Teams are not talent pools in disguise at the disposal of line managers.

Remedy: Scrum Masters should not restrict stakeholders' access to team members, but they should educate stakeholders on how to best communicate with the Scrum Team to foster an effective relationship. Some suggestions on how to proceed are as follows:

- **Set boundaries:** Establish clear guidelines about when and how stakeholders can interact with the Scrum Team, aligning with the Scrum Guide's assertion of minimizing disruptions.[3]
- **Educate stakeholders:** Educate stakeholders on the impact of disruptions on the Scrum Team's productivity and how they jeopardize the Sprint Goal.
- **Assert team autonomy:** Uphold the principle that Scrum Teams manage their composition and workflow, underscoring their right to self-organize. (Advocating for this principle proves challenging in many organizations.)
- **Preserve Scrum events:** Protect the integrity of Scrum events, emphasizing their importance in supporting team coordination and communication, not for reporting purposes.
- **Strengthen Product Owners:** Emphasize the Product Owner's role as the first line of communication between stakeholders and the Scrum Team to maintain focus.

11. Defining Technical Solutions

Observation: An engineer turned Scrum Master is now "suggesting" how the Developers implement issues. Alternatively, the Product Owner or an outsider, such as a technical lead, is pursuing the same approach.

Background: Why might a Scrum Master overstep the boundaries of their role and get involved in implementation questions when the Scrum Guide clearly states that any decision on how to turn a Product Backlog item into a done Increment is solely the decision of the Developers? Some reasons may include:

- **Old habits:** The Scrum Master, an engineer in the past, may naturally gravitate toward providing solutions based on their technical experience. They may

3. Check out Jimmy Janlén's interruption buckets in *Toolbox for the Agile Coach: 96 Visualization Examples: How Great Teams Visualize Their Work* (Crisp Publishing, 2015).

inadvertently revert to their old role out of habit or believe they are genuinely helping the Developers.

- **Underestimation of team skills:** The Scrum Master may underestimate the Developers' capabilities, believing that the team needs their guidance in technical matters. This perception could stem from a lack of trust or unfamiliarity with the team's skills and competencies.
- **Pressures and self-preservation:** If management or stakeholders pressure the Scrum Master to deliver specific results, they might feel compelled to interfere in the Developers' work. They could think their role is on the line if the team doesn't meet these expectations.
- **Control issues:** The Scrum Master, because of personal insecurities or a perceived lack of progress, may feel the need to control the process. They may believe they can speed up the process by directing the work rather than allowing the Developers to self-manage.
- **Inadequate transition:** Transitioning from a technical role to a Scrum Master is a significant shift. It requires a change in mindset from being a doer to becoming an enabler. If this transition is not adequately supported or understood, the Scrum Master might fall back into the comfort zone of their previous role.
- **Fear of obsolescence:** The Scrum Master may worry that by not involving themselves in the technical decisions, they will become disconnected from the actual work, which could make them feel less valuable or even obsolete.

Remedy: So, what can we do about this Scrum Master anti-pattern? How can we convince the Scrum Master to let go of their former self as a Developer and doer and embrace the role of an enabler? Consider the following:

- **Open dialogue:** Have a frank conversation, perhaps during a Retrospective, about the Scrum Master's role. Highlight the principles of the Agile Manifesto and the Scrum Guide, emphasizing that the Scrum Master's role is to facilitate, not dictate, the implementation. Discuss the benefits of self-organizing teams and how they improve team productivity and creativity.
- **Training and education:** Encourage the Scrum Master to attend further training or workshops to better understand the nuances of their role. It's beneficial to learn from the experiences of other Scrum Masters who have transitioned from technical positions.

- **Peer feedback:** Encourage the Developers to provide feedback whenever they feel the Scrum Master is overstepping their boundaries. They should give this feedback constructively, emphasizing how such behavior affects the team's productivity and autonomy.
- **Role modeling:** Share examples of successful Scrum Masters who have effectively transitioned from a technical role, showing the Scrum Master that stepping back from implementation does not diminish their value, reputation, or standing.
- **Mentoring or coaching:** A more experienced Scrum Master or agile coach can guide the transitioning Scrum Master, providing them with strategies to combat the urge to delve into technical details.
- **Self-reflection:** Encourage the Scrum Master to practice self-reflection. It might be helpful for them to question why they need to involve themselves in the implementation and how they can work toward overcoming this inclination.

THE DAILY SCRUM

12. Not Supporting Struggling Developers

Observation: A Developer experiences difficulties accomplishing an issue over several consecutive days, and nobody on the Scrum Team offers help. Moreover, the Scrum Master fails to facilitate the necessary discussion.

Background: There may be many reasons for a Developer's struggle. Here are just a few:

- **Task complexity:** The task may be more complex or challenging than initially anticipated, for example, due to technical debt.
- **Lack of experience or skills:** The Developer might not have the necessary skills or experience to complete the task.
- **Health or personal issues:** Personal or health issues may impact the Developer's performance.
- **Fear of criticism or failure:** The Developer might be reluctant to admit difficulties due to fear of judgment or criticism.
- **Lack of team collaboration:** The Developer workload has reached such an unproductive level that they no longer can support each other.

The worrying issue is less the struggle than the lack of support by the teammates and, more important, the Scrum Master.

Remedy: There are many ways to address the preceding issues, from improving craftsmanship to employing an effective Definition of Done to regular skill-sharing sessions to creating a safe space where no one is judged but is supported.

However, regarding the failure of the Scrum Master to detect the necessity to support a team member, I suggest using a simple visualization technique from Kanban: the "work item age." It is about tracking and displaying the time a Product Backlog item has spent in a specific state, such as "in development." This time is often represented visually on the Kanban board, for example, by adding a marker for each day the item did not move. Once an item accumulates a predefined number of markers, the team offers support.[4]

It is helpful to agree on this practice as a part of the team working agreement to create a shared understanding among all team members that they need to support each other.

13. Not Preventing Stakeholders from Attendance

Observation: Although the Developers decided to limit attendance to the Daily Scrum to the Scrum Team members, stakeholders show up uninvited, and the Scrum Master fails to address the issue.

Background: It is the Developers' prerogative to decide how to run the Daily Scrum. If they choose to keep stakeholders away from it, this decision needs to be respected by everyone else.

Following are possible reasons that Scrum Masters fail to address the inappropriate behavior of the stakeholders attending the Daily Scrum uninvited:

- **Inadequate support from the organization:** Without enough backing from the organization, the Scrum Master might find it challenging to address the issue, particularly in organizations that consider their role to be team-focused.

4. Yuval Yeret, "4 Key Flow Metrics and How to Use Them in Scrum's Events," *AgileSparks,* May 10, 2018, https://medium.com/agilesparks/4-key-flow-metrics-and-how-to-use-them-in-scrums-events-c63a5a2e1bcf.

- **Conflict avoidance:** They may want to avoid conflicts or uncomfortable stakeholder conversations.
- **Lack of confidence:** The Scrum Master may hesitate to confront stakeholders higher up in the organization. (Ask yourself: Would you show Elon Musk the door?)
- **Ineffective communication skills:** The Scrum Master may lack the communication skills necessary to deal with high-level stakeholders.
- **Stakeholder pressure:** There could be high pressure from stakeholders on the team to perform better, making it difficult for the Scrum Master to intervene.
- **Fear of repercussions:** The Scrum Master might fear negative repercussions, like job loss or other political consequences within the organization.

Remedy: There are several ways a Scrum Master can educate stakeholders who show up uninvited to the Daily Scrum and explain that they need to stop this behavior unless the Developers invite them:

- **Set clear boundaries:** Communicate the rules and expectations for the Daily Scrum, emphasizing that it is an event for and by the Developers and the need to respect their autonomy.
- **Provide education:** Teach stakeholders about the purpose and structure of the Daily Scrum, explaining why their presence could disrupt the event.
- **Stakeholder meetings:** If necessary, arrange separate meetings for stakeholders to voice their concerns and queries, ensuring they feel heard without interrupting the Daily Scrum.
- **Use visual aids:** Use signs or visual cues indicating that the Daily Scrum is in session and interruptions are not welcome unless invited. (An "On Air" sign works for Scrum Teams, too.)
- **Encourage self-management:** Foster the Developer's self-esteem, enabling them to address the issue directly with the stakeholders.
- **Involve management:** If necessary, involve higher management to support Scrum's rules and the Scrum Master's efforts.

THE RETROSPECTIVE

The final set of Scrum Master anti-patterns addresses the Sprint Retrospective.

14. Groundhog Day

Observation: The Sprint Retrospective never changes in composition, venue, or length.

Background: In this case, the Scrum Team tends to revisit the same issues repeatedly—it's Groundhog Day without the happy ending. It has become a boring exercise, a ritualized formality—what was good during the last Sprint, what was bad, let's have some action items—that proves of little and, in time, diminishing value to the Scrum Team.

Remedy: Try to vary the format of the Retrospective; for example:

- Go meta and have a Retrospective on your Retrospective.
- Have a themed Retrospective.[5]
- Change the venue or the time.
- Disguise the Retrospective as a team breakfast.

There are so many things a Scrum Master can do to make Retrospectives great again and reduce the absence rate. Be creative with every single Retrospective and aim not to repeat a single format or theme. Liberating Structures and many other websites dedicated to organizing Retrospectives make this an affordable and worthwhile investment.

Toolbox Tip

Retromat.org is a free website by Corinna Baldauf dedicated to helping you tailor Retrospectives to your Scrum Team's needs: "Planning your next agile retrospective? Start with a random plan, change it to fit the team's situation, print it and share the URL. Or browse around for new ideas!"

15. Let's Have the Retro Next Sprint

Observation: The Scrum Team postpones the Retrospective to the next Sprint.

See Chapter 9, anti-pattern 4, for a detailed description.

5. Use themed Retrospectives to be creative as a team. For many Westerners, for example, Halloween may come to mind. Other topics may include your favorite video game, song, movie, opera, or book. What about travel, gardening, or cooking? There is no limit to your creativity. (Pro tip: Organize your themed Retrospective during a team-building event.)

16. #NoRetro

Observation: There is no Retrospective, as the team believes there is nothing to improve, and the Scrum Master accepts this notion.

See Chapter 9, anti-pattern 1, for a detailed description.

17. #NoDocumentation

Observation: No one is taking minutes for later use.

See Chapter 9, anti-pattern 10, for a detailed description.

18. The Blame Game

Observation: The Retrospective is an endless cycle of blame and finger-pointing.

See Chapter 9, anti-pattern 14, for a detailed description.

19. No Psychological Safety

Observation: One or two team members dominate the Retrospective.

Background: This communication behavior is often a sign of a compromised environment that is no longer psychologically safe.

Retrospectives support a Scrum Team's path to excellence only when everyone can address issues and provide their feedback free from third-party influence. If some team members dominate the conversation and perhaps even bully or intimidate other teammates, the Retrospective will fail to provide such a safe place. This failure will result in participants dropping out of the Retrospective, rendering the results less valuable.

Remedy: It is the primary responsibility of the Scrum Master, in their role as a chief facilitator,[6] to ensure that everyone is heard and has an opportunity to voice their thoughts.

6. I believe that every Scrum Team member needs to be able to run any Scrum event, Retrospectives included. Of course, the Scrum Master will likely be the most skilled facilitator. However, that does not mean the Scrum Master must be the default facilitator.

By the way, equally distributed speaking time is, according to Google, also a sign of a high-performing team. Learn more: What Google Learned from Its Quest to Build the Perfect Team.[7]

If a Scrum Master needs to restore psychological safety, several angles may prove successful:

- Set clear expectations for Retrospectives by pointing to Scrum Values, emphasizing the importance of respectful communication, active listening, and equal participation. Encourage the team to hold each other accountable to these rules.
- Seek to speak with the dominant individuals privately, addressing their behavior and its impact on the team. Help them understand the importance of psychological safety, and request their support in creating a more inclusive environment.
- During Retrospectives, watch the dynamics and intervene when necessary to maintain a balanced and inclusive discussion. Redirect the conversation if it becomes unproductive or if specific individuals start to dominate or bully others.
- Utilize structured discussion formats, such as round robin or Conversation Café, where each team member takes turns sharing their thoughts. These practices can help ensure that everyone has an opportunity to speak and that the conversation remains balanced.
- Actively involve introverted or less vocal team members by asking for their opinions, prompting them to share their thoughts, and acknowledging their contributions. Offer support and reassurance when necessary. However, please note that not all of your less vocal teammates may feel comfortable with this approach.
- Offer different ways for team members to share their thoughts and feedback, such as written input or anonymous submissions and surveys. Anonymity encourages everyone to share their perspectives, even if they are uncomfortable speaking up during the Retrospective.
- Inspect the effectiveness of Retrospectives and adapt as needed. Gather feedback from the team on how the Retrospectives work, and implement changes to improve psychological safety.

7. Charles Duhigg, "What Google Learned from Its Quest to Build the Perfect Team," *The New York Times,* February 28, 2016.

(See also the Remedy section of anti-pattern 14, The Blame Game, in Chapter 9.)

20. Excluding Stakeholders All the Time

Observation: The Scrum Team categorically rejects having Retrospectives with their stakeholders.

See Chapter 9, anti-pattern 5, for a detailed description.

FOOD FOR THOUGHT

Agile Coaching by Scrum Master? How come Scrum Masters are often stuck in a tactical role at the team level when we all know that a successful Scrum Master increasingly shifts their focus from the Scrum Team to the organization? Or is this effect just collateral damage of a well-known scaling framework's classification of the Scrum Master role?

Scrum at the team level is not rocket science; however, the serious impediments to becoming agile often reside at the organizational level. So why does it seem necessary to create a new role—the agile coach or organizational coach—to address this need?

Redundancy: Is a Scrum Master of a successful team redundant at the team level? Can they consequently move to a pool of Scrum Masters from which a team needing support can request assistance?

CONCLUSION

There are plenty of possibilities to fail as a Scrum Master. Sometimes, it is the lack of organizational support. Some people are not suited for the job. Others put themselves above their teams for questionable reasons. Moreover, some Scrum Masters lack feedback from their Scrum Teams and stakeholders. Whatever the case, try to lend your Scrum Master in need a hand to overcome the misery. Scrum is a team sport.

Toolbox

Regarding the following tips and exercises, you will find detailed descriptions in Appendix B:

- **"20 Questions from a New Scrum Master to the Scrum Team"** comprises 20 questions that fit into a 60-minute timebox, from Product Backlog forensics to metrics to team challenges and technical debt.
- **"20 Questions from a New Scrum Master to the Product Owner"** comprises 20 questions addressing the future collaboration between the two individuals and the rest of the Scrum Team.
- **"20 Questions from a New Scrum Master to the Developers"** comprises 20 questions addressing the foundations of a Scrum Team's capability to build valuable products: from technical excellence and what it takes to achieve this high proficiency level to technical debt.
- **"Agile Metrics—The Good, the Bad, and the Ugly"** points to suitable agile metrics reflecting either a Scrum Team's progress in becoming agile or an organization's progress in becoming a learning organization.
- **"Building Communities of Practice"** helps win hearts and minds within the organization. It provides authenticity to the agile transformation—signaling that the effort is not merely another management fad.
- **"How to Create a Definition of Done"** details a collaborative exercise of the whole Scrum Team, probably including some of the stakeholders at a later stage, to create, inspect, and adapt a Definition of Done.
- **"Meta-Retrospectives—How to Get Customers and Stakeholders Onboard"** details an excellent Retrospective format to foster collaboration within the extended team, including your team's stakeholders, to build a shared understanding of the big picture and create valuable action items.
- **"Peer Recruiting"** is a guide on how to hire Scrum Masters (and other agile practitioners) by including self-managing teams in the decision process.
- **"Retrospectives with Distributed Teams"** describes a string of Liberating Structures microstructures that distributed Scrum Teams can use to organize meaningful Retrospectives.
- **"Stakeholder Communication Tactics"** comprise eleven proven stakeholder communications tactics to help you not just do Agile but *become* agile.

This page intentionally left blank

2 Product Owner Anti-Patterns

Introduction

No other role in Scrum can contribute to mediocre outcomes like the Product Owner can—garbage in, garbage out. If they don't understand their customer's needs, they will make bad decisions about where the Scrum Team should spend its time. They can reduce this risk by collaborating and aligning with their teammates on the Product Goal and the Product Backlog; see Figure 2.1.

Figure 2.1 No Product Owner should be a loner; creating value for customers is a (Scrum) team sport.

THE ROLE OF THE PRODUCT OWNER ACCORDING TO THE SCRUM GUIDE

The Scrum Guide characterizes the job of the Product Owner as follows:

- Maximizing the product's value resulting from the Scrum Team's work
- Developing and explicitly communicating the Product Goal
- Managing the Product Backlog, including delegating responsibilities while remaining accountable
- Communicating with stakeholders and representing the needs in the Product Backlog through direct communication or Scrum events such as the Sprint Review
- Ensuring that the Product Backlog is transparent, visible, and understood
- Ordering Product Backlog items
- Defining business objectives for Sprints
- Canceling Sprints if a Sprint Goal becomes obsolete[1]

Most Product Owner anti-patterns result when Product Owners misunderstand their Product Backlog management responsibilities. Less successful Product Owners also fail to communicate with teammates and stakeholders.

When Product Owners fall short, it isn't typically because they lack project management skills, cannot maintain product requirements documents, or fail to master a tool used to manage the Product Backlog. Successful Product Owners spend less time working on updating the Product Backlog and more time understanding customer needs and creating alignment between the Scrum Team and Stakeholders; managing the Product Backlog is simply a means to creating this alignment. Successful Product Owners spend time creating a transparent system that identifies potentially valuable[2] ideas from a variety of sources, relentlessly communicate the

1. Ken Schwaber and Jeff Sutherland, "Product Owner," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#product-owner.
2. *Product discovery* is a strategic process in which product managers explore and determine valuable solutions. They aim to align users' needs and the organization's profit goals, leading to a solid return on investment. The process involves researching, ideating, validating, and testing potential product features or enhancements before building them. Consider reading Marty Cagan's *Inspired: How to Create Tech Products Customers Love,* 2nd ed. (Wiley, 2018) for a deeper understanding.

customer's needs, and collaborate with the Developers to deliver a solution that satisfies those needs.

PRODUCT BACKLOG AND REFINEMENT ANTI-PATTERNS

You can spot many Product Owner anti-patterns in the Product Owner's backyard—the Product Backlog and its refinement:

1. Part-Time Product Owner

Observation: The Product Owner is not working daily on the Product Backlog.

Background: The Product Backlog needs to represent the best use of the Developers' capabilities and time at any given moment. If the Product Backlog is not effectively refined, the Scrum Team may not end up working on the most valuable things.

The Product Backlog's contribution to value creation derives, in a significant part, from open discussion about value between the Product Owner and the Developers. These discussions often result from questions about value that the Product Owner and Developers must answer for themselves, such as the following:

- Does a new product idea make sense to other team members? Is it really the best use of the Developers' time?
- Is the Product Owner pursuing an idea based on a lack of technical understanding that the Developers could alleviate?
- Could we strip down the new feature idea to a less expensive level for the first version?
- What technical tasks do the Developers need to address to preserve the quality of the underlying technology stack?

Remedy: When teams are effective, the Developers constantly challenge the Product Owner regarding the value of suggested Product Backlog items. These challenges reduce the probability that the Product Owner will exhibit confirmation bias and help the Scrum Team to make better decisions about what to work on when they plan the Sprint. As the saying goes: Love the customers' problem, not your solution.

When the Product Owner is not fully available to work with the team, they struggle to know where they should spend their time. They may even work on things that have low value, which reduces the return the organization makes on their effort, time, and money.

2. Copy & Paste Product Owner

Observation: The Product Owner creates Product Backlog items by breaking down requirements documents received from stakeholders into smaller chunks.

Background: This scenario helped to coin the nickname "ticket [animal of your choice]" for Product Owners because the Product Owner does little more than take input from stakeholders and create a ticket (i.e., a Product Backlog item.)

This pattern results from a fundamental misunderstanding of the role of the Product Backlog and the importance of Product Backlog refinement: Stakeholders, like everyone, can be wrong. Merely recording their requests misses an opportunity to better understand the need behind the stakeholder request. Engaging the entire team in this critical analysis helps everyone to gain a shared understanding of the real need. These discussions also help to create alignment between team members that mitigates the risk of working on low-value Product Backlog items.

Product Backlog refinement is important because it stimulates these discussions. When it is skipped because the Product Owner regards the Product Backlog as simply a warehouse of stakeholder requests, the Scrum Team misses important opportunities to improve the value they deliver.

Remedy: Your Scrum Team needs to talk about Product Backlog items during refinement. This process requires discourse that "copy & paste" will not provide. Think of Karl Popper: "Always remember that it is impossible to speak in such a way that you cannot be misunderstood."[3] There are several reasons for that effect; for example:

Ambiguity: Language can be inherently ambiguous, leading to different interpretations. Two people can read the same sentence and come away with different meanings.

3. Goodreads, Karl Popper quotes, https://www.goodreads.com/quotes/7675070-always-remember-that-it-is-impossible-to-speak-in-such.

Context misalignment: Developers and stakeholders often operate in different contexts, so their understanding of a term or concept can differ.

Missing tacit knowledge: Written documents may not convey the implicit knowledge, assumptions, or intent behind the requirement.

Evolution of needs: Needs and requirements evolve. A static document may not reflect current realities.

Lack of shared understanding: Conversations foster a shared understanding through interaction and mutual learning, which is hard to achieve through documents.

Consequently, as a first step, I recommend to start writing a Product Backlog item together as a Scrum Team in a refinement session; see Figure 2.2.

Practical Tales: *In my time as a Product Owner, I never wrote entries to the Product Backlog. Instead, I prepared a pitch for my teammates about why I considered an issue a worthy Product Backlog item. For example, I brought data and told a compelling story.*

Once we agreed on the usefulness of an item, a Developer wrote the Product Backlog item. I made a great effort to ensure that the resulting item was a team issue, our issue, not a Product Owner issue.

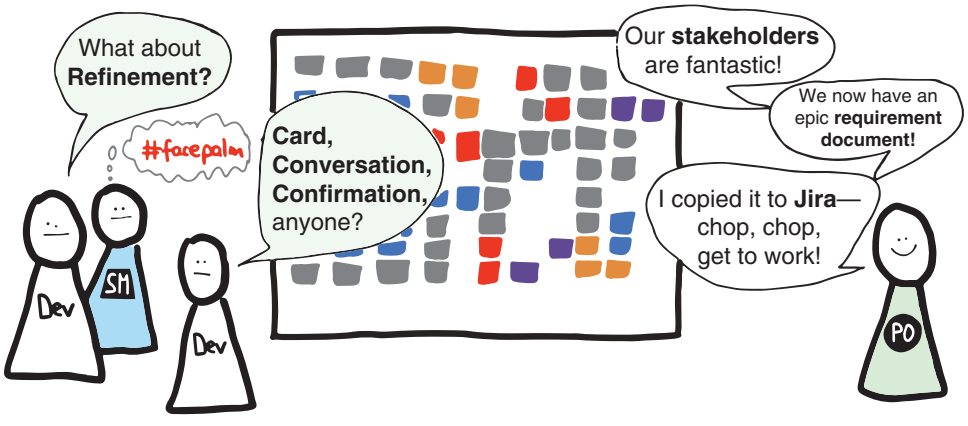

Figure 2.2 Product requirements documents are one of many sources, not the source of information for Product Owners.

3. Storage for Ideas

Observation: The Product Owner uses the Product Backlog as a repository of ideas and requirements.

See Chapter 10, anti-pattern 11, for a detailed description.

4. The Overreaching Product Owner

Observation: The Product Owner creates Product Backlog items by not just providing the why but also defining the what and the how.

Background: The division of labor in a Scrum Team is simple:

- The Product Owner answers the why question: Why is the Product Backlog item a valuable investment that helps solve the customers' problem?
- The Developers and the Product Owner collaborate on the what question: What scope is necessary to achieve the desired impact?
- The Developers answer the how and the who questions: What will the technical implementation look like, and who might be interested in working on it?

There are a few reasons why a Product Owner feels motivated to answer the how question; for example:

- They are working with a junior Scrum Team and have significantly better knowledge of the technology stack than the Developers have.
- Before moving to the Product Owner role, they were a Developer on the team.
- They lack trust in the Developers' capabilities, for example, because of stakeholder feedback or earlier experiences.
- They try to be supportive in good faith.

Remedy: Let the people closest to the job at hand figure out how to do it; in a complex environment, there are no experts: "How this is done is at the sole discretion of the Developers. No one else tells them how to turn Product Backlog items into Increments of value."[4]

4. Schwaber and Sutherland, "Product Owner."

Guiding by objectives and providing support when requested or needed will be appreciated, though.

Additionally, you may want to run a Retrospective to gain insight into the motivation behind this step.

5. Prioritization by Proxy

Observation: A single stakeholder or a committee of stakeholders decides on the nature of the Product Backlog. The Product Owner liaises for this or these individuals, passing their ideas to the rest of the team.

See Chapter 10, anti-pattern 1, for a detailed description.

6. 100% in Advance

Observation: The organization tasks the Scrum Team with re-creating a legacy application one-for-one on a new technology base. The Scrum Team therefore prepares the complete Product Backlog before they head for the first Sprint. After all, it is the same set of functionality.

See Chapter 10, anti-pattern 3, for a detailed description.

7. Oversized Product Backlog

Observation: The team creates a massively comprehensive Product Backlog.

See Chapter 10, anti-pattern 2, for a detailed description.

8. Outdated Items

Observation: The Product Backlog contains items that haven't been touched for months.

See Chapter 10, anti-pattern 4, for a detailed description.

9. Everything Is Detailed and Estimated

Observation: All Product Backlog items are thoroughly detailed and estimated.

See Chapter 10, anti-pattern 5, for a detailed description.

10. Missing Acceptance Criteria

Observation: There are Product Backlog items that need additional acceptance criteria.

See Chapter 10, anti-pattern 7, for a detailed description.

11. Crowding-Out Collaboration

Observation: The Product Owner invests too much time upfront in creating Product Backlog items, making them too detailed.

Background: If a work item looks complete, the Developers might not see the necessity to get involved in further refinement.

Following are possible reasons for this behavior:

- **Lack of ownership:** The team members may not feel the need to contribute or take ownership, as the perception of a "finished" Product Backlog item may reduce their intrinsic motivation to actively participate.
- **Fear of interfering:** Team members may hesitate to question or critique a seemingly complete solution; they may fear being seen as disruptive or uncooperative. This fear can prevent individuals from raising concerns or offering valuable input.
- **Lack of psychological safety:** If the team's environment does not foster psychological safety, team members may be reluctant to voice their opinions, even if they believe the issue needs further collaboration or refinement.

Consequently, when the Product Owner presents a solution—even unintentionally—as complete, it may limit the space for creative thinking and problem-solving. The team might miss opportunities to explore alternative ideas or innovations, as they may feel that the presented solution is the only viable option.

This way, a "fat" Product Backlog item reduces the team's engagement level, compromising the creation of a shared understanding, possibly reducing value creation.

Remedy: If your Product Owner tends to present "complete" Product Backlog item at the beginning of a refinement session, reach out to them and ask whether they may be open to run an experiment: for their own purpose, they may come up with their usual, complete Product Backlog item draft. However, at the team level, they

should only provide an outline, supported by data, insights, acceptance criteria, and a compelling story on why this item is valuable.

Once the Product Backlog item is collaboratively created, compare both versions and take it from there.

Product Owners should like the idea of collaborating on Product Backlog items, as it is a valuable exercise that saves them time, too!

By the way, we didn't witness this effect back in the days when we used index cards, given their physical limitation.

12. The "I Know It All" Product Owner

Observation: The Product Owner does not involve stakeholders or subject matter experts in the refinement process.

Background: A Product Owner who believes they are omniscient or who is a communication gateway is a risk to the team's success. Capable teams, in general, and Product Owners, in particular, know well when to reach out to others to better understand a matter at hand.

There could be several reasons why this anti-pattern occurs; for example:

- It is possibly a sign of inexperience, ignorance, or overconfidence if a Product Owner never asks for support from others to improve their knowledge of the problem and solutions space.
- Others in the organization might consider asking for help a sign of weakness or incompetence with career-limiting consequences.
- The Product Owner likely is not aware that others might support the team's efforts and does not know who those people are.

Remedy: Generally, whenever the Scrum Team feels the need to understand a problem space better, reach out to your stakeholders and customers for support or consider hiring a subject matter expert to shed light on the issue.

It may also pay to run a Retrospective to address this particular topic. Conducting an anonymous survey in advance can provide additional insight, particularly in an environment with little psychological safety.

13. No Research

Observation: The Product Backlog does not contain any, or contains only a few, timeboxed research tasks, or spikes.[5]

Background: Figuring out what is worth building is often an incremental process. It may start with quantitative or qualitative data that points to an interest among stakeholders and customers in certain functionality. For example, users unexpectedly utilize an existing feature to improvise a workaround to substitute for a nonexistent feature. This learning is a good starting point but requires more attention.

You may want to run user interviews to understand the problem space better. Moreover, before committing your team to build a capability based on a hunch that users will appreciate that feature, you want to develop a few prototypes to validate your assumptions. Once you validate your assumptions, the Developers might be interested in exploring different technical ways of implementing this functionality.

Neither of these activities creates any value for stakeholders or the organization by themselves. Instead, they are necessary steps to help decide to add an item to the Product Backlog. As these steps require attention from team members, reducing the capacity of the Scrum Team to work on other issues, it is a good practice to be transparent and add them to the Product Backlog.

Remedy: There are several approaches to the issue. If your team already does research work during Sprints, the simple fix is to visualize those tasks in the Product Backlog.

However, if the Scrum Team spends too much time discussing future features during Product Backlog refinement instead of researching them with spikes, you should switch gears: pick a Sprint and run an experiment on running spikes.

If your stakeholders do not expect your Scrum Team to run any research, consider contacting them. Help them understand how valuable it is to the Scrum Team to

5. Some people also refer to them as "spikes," a term from Extreme Programming. A spike is a timeboxed exploration or investigation to reduce uncertainty or risk surrounding a technical issue or to answer a specific question. The purpose of a spike is to gather knowledge or to test the feasibility of a solution before committing to its implementation. Learn more about spikes here: Kent Beck, *Extreme Programming Explained: Embrace Change*, 2nd ed. (Addison-Wesley, 2004).

gain first-hand insight into your customers' problems and possible solutions. It helps to align the Scrum Team with the big picture, thus increasing the chance of value creation.

14. Last-Minute Product Backlog

Observation: The Product Owner and the rest of the Scrum Team do not invest adequately in Product Backlog creation and refinement. Consequently, they rush to fill the Product Backlog immediately before the Sprint Planning to ensure there is enough work for the upcoming Sprint to meet stakeholder and management expectations.

Background: Garbage in, garbage out. Scrum Teams who follow this approach fundamentally misunderstand the purpose of Scrum, including when to use it and why: The purpose of Scrum is not to keep the team occupied but to help a team solve the most urgent and rewarding customer problems. Consequently, Scrum is outcome-focused, not output-focused.

Maximizing the utilization rate of the Scrum Team members or keeping them "busy as we pay them" is not the purpose of the exercise.

There is no correlation between building more code and creating more value.

Remedy: The Scrum Team—and particularly the Product Owner—need to embrace the idea that the Product Backlog is a critical artifact to a team's success. The Product Backlog should reflect the best use of the team's time at any given moment. To keep the artifact at that level requires the team to continuously refine the Product Backlog.

15. Involving the Scrum Team—Why?

Observation: The Product Owner does not involve the entire Scrum Team in Product Backlog refinement and instead relies on just the "lead engineer" and a designer. Moreover, the Developers are okay with this approach, as it allows them to write more code—an activity they prefer over figuring out what is worth building.

Background: There are no experts but many competing ideas when trying to solve a complex problem. Therefore, limiting the active participants in refinement activities to a few team members instead of the whole Scrum Team increases the risk of falling victim to confirmation bias as the diversity of opinion is artificially limited.

Remedy: Not all team members must participate in refining all Product Backlog items. However, everyone should have the opportunity to express their interest in refining a particular Product Backlog item.

For that purpose, I recommend having at least one Product Backlog refinement session per Sprint that all Scrum Team members attend to create alignment on who will be working on what. Those interested in participating in item-specific refinement work can organize future sessions at their discretion.

SPRINT PLANNING ANTI-PATTERNS

The number two area on my list of Product Owner anti-patterns is the Sprint Planning itself.

16. The Scrum Team Has No Sprint Goal

Observation: The Product Owner proposes a list of Product Backlog items for the Sprint that is simply an arbitrary assortment of tasks without cohesion. Consequently, the Scrum Team cannot create a Sprint Goal.

See Chapter 14, anti-pattern 1, for a detailed description.

17. The Scrum Team Does Not Respect Sprint Boundaries

Observation: The Scrum Team habitually takes on too many tasks and moves unfinished work directly to the next Sprint, as they would when using Kanban.

See Chapter 14, anti-pattern 6, for a detailed description.

18. Last-Minute Changes

Observation: The Product Owner tries to squeeze in some last-minute Product Backlog items that have not been refined.

See Chapter 6, anti-pattern 10, for a detailed description.

19. Output Focus

Observation: The Product Owner pushes the Developers to take on more tasks than they can realistically handle. The Product Owner may express this in terms of team metrics such as velocity to support their desire.

Background: The purpose of Scrum is not to maximize the amount of work the Developers can perform. It is to solve customer problems within the given time constraints while contributing to the organization's sustainability.

Product Owners push the Scrum Team to maximize output for a variety of reasons, including:

- **They misunderstand velocity:** Velocity is a measure of the amount of work a team can tackle during a single Sprint and is unique to each team. Misusing it as a team performance metric usually pushes the team to increase their velocity and output, but not necessarily the value they deliver to customers.
- **They are pressured by stakeholders to increase output:** Stakeholders can pressure the Product Owner to deliver more features or functionality in a shorter timeframe.
- **They are pressured by deadlines:** Imminent business-imposed deadlines may cause the Product Owner to push the Developers to produce more work, including launch dates, marketing campaigns, and quarterly financial goals.
- **They are pressured by competitors:** The market competition is fierce, and the company is trying to outdo its competitors.
- **They don't understand or accept the importance of sustainable pace:** The Product Owner may need to fully understand the principle of sustainable pace, which states that teams should work at a pace they can sustain indefinitely without causing burnout. They may mistakenly believe that pushing for more output will result in more productivity.
- **Their organization measures team performance based on output:** In some organizations, the value or effectiveness of a Scrum Team might be based on how much the team produces, not how much value it delivers.

Pushing for output maximization is the road to becoming a feature factory.[6] Moreover, it violates not only the Developers' prerogative to pick Product Backlog items for the Sprint Backlog but also Scrum Values in general.

6. John Cutler's "feature factory" refers to a software development organization that focuses purely on output, typically measured in the number of features or user stories delivered, without considering those features' actual impact or value. A feature factory treats software development like an assembly line, emphasizing speed and volume instead of thoughtful creation. Learn more: John Cutler, "12 Signs You're Working in a Feature Factory," @johncutlefish's blog, November 17, 2016, https://cutle.fish/blog/12-signs-youre-working-in-a-feature-factory.

Remedy: Scrum's purpose isn't to maximize output—it's to deliver the most value possible in a sustainable way. It's crucial to convey this message clearly to the Product Owner:

- **Educate the Product Owner and stakeholders:** If the Product Owner and/or stakeholders misuse velocity as a productivity measure, educate them about using velocity as a forecasting tool, not a performance metric.
- **Facilitate communication:** Help the Product Owner communicate with stakeholders and manage their expectations. Encourage the Product Owner to be transparent about what the team can realistically achieve in a given Sprint, pointing to the impact of overloading the Scrum Team.
- **Advocate for sustainable pace:** Explain the importance of maintaining a sustainable pace for the team's long-term health and productivity. Discuss the concept of technical debt and the potential consequences of burnout.
- **Encourage reflection:** Use Retrospectives as an opportunity to address these issues. Facilitate a conversation where the Developers can share their feelings about the workload, and the Product Owner can explain their perspective. The goal is to find a middle ground that respects the team's capacity while meeting business needs.
- **Provide support in capacity planning:** Assist the Product Owner in understanding the team's actual capacity.
- **Mentor the Product Owner on empirical process control:** Teach the Product Owner about empiricism, the decision-making process based on what is known. Encourage them to adjust the scope based on the observed team's velocity rather than forcing the Scrum Team to meet a predetermined scope.

SPRINT ANTI-PATTERNS

Another area prone to Product Owner anti-patterns is the Sprint.

20. Absent Product Owner

Observation: The Product Owner is absent for most of the Sprint and is not available to answer questions from the Developers regarding Product Backlog items from the Sprint Backlog.

Background: As the Sprint Backlog is emergent, there are multiple reasons why the Developer may want to talk to the Product Owner; for example:

- Developers may identify necessary new work.
- The scope of previously identified work may need to be adjusted.
- A refined Product Backlog item turns out to be significantly larger than initially considered.
- The Developers run into previously unknown technical debt that won't allow them to proceed as planned and want to align with the Product Owner on the news approach.
- A Developer wants to show the Product Owner a Product Backlog item from the Sprint Backlog they consider done.

Whatever the case, the Product Owner's absence will likely leave the Developers in the dark, creating undone work that requires a decision on how to move on. No matter the reasons for the Product Owner's absence, it displays the team's value creation, putting the accomplishment of the Sprint Goal at risk. It is an expensive Sprint Backlog anti-pattern.

Remedy: Help the Product Owner understand that their contribution to the Scrum Team's success requires spending significant time with their teammates. If they struggle with their time budget, for example, because the organization's expectations leave them overextended, try to support them to free time for interacting with their teammates.

A good candidate for this approach is Product Backlog management; your Product Owner does not need to create all entries by themselves. Moreover, other options to support the Product Owner are in product discovery, such as creating prototyping or running user interviews.

21. Product Owner Changes Work Items in the Sprint Backlog during the Sprint

Observation: The Product Owner cannot let go of Product Backlog items once they become part of the Sprint Backlog. For example, the Product Owner increases the scope of a work item or changes acceptance criteria once the Developers select this Product Backlog item for the Sprint Backlog; see Figure 11.3.

See Chapter 11, anti-pattern 14, for a detailed description.

22. Delaying Product Owner

Observation: Despite the Developers inquiring, the Product Owner does not provide feedback on Product Backlog items once they are done. Instead, they wait until the end of the Sprint to review all done work.

Background: The Scrum Team benefits when the Product Owner provides immediate feedback on done Product Backlog items because it lets them address issues that might otherwise prevent them from achieving their Sprint Goal.

Some flawed reasons for that behavior might be:

- **Time constraints:** Product Owners may delay feedback due to time constraints, often juggling multiple responsibilities and commitments.
- **Prioritizing stakeholder interactions:** Some Product Owners may prioritize stakeholder interactions and other nondevelopment activities over reviewing done work.
- **Cultural habits:** Some Product Owners, especially those from more traditional backgrounds, may feel more comfortable waiting until the end of the Sprint to review work.

Providing feedback on Product Backlog items is not a work acceptance or quality gate (there is no such thing in Scrum). Once a Product Backlog item meets the Definition of Done, it can be released into the hands of users.

Nevertheless, seeking and providing feedback has many advantages; for example, strengthening team collaboration, reducing work-in-progress, thus supporting a sustainable pace of work, and improving the probability that the Scrum Team accomplishes the Sprint Goal.

Remedy: Convince the Product Owner to provide feedback on work items as soon as they are done. Here are some ideas worth trying:

- **Educate the Product Owner on the impact of feedback:** Highlight the impact of feedback on the Developers' productivity and morale, helping them to improve their delivery capability and forecast quality.
- **Stress the importance of inspection and adaptation:** Explain to the Product Owner that timely feedback allows the team to improve Increments quickly.

- **Encourage a culture of ongoing collaborative reviews:** Explain to the Product Owner that providing early feedback helps the Scrum Team to be more responsive.
- **Prioritize providing feedback:** Work with the Product Owner to prioritize and allocate time for immediate feedback.
- **Remove impediments:** If organizational issues are the root cause of the delay in feedback, work to remove these impediments.

23. Sprint Stuffing

Observation: The Developers accomplished the Sprint Goal early, and the Product Owner is pushing them hard to accept new work from the Product Backlog to fill the "void" of the Sprint Backlog.

See Chapter 11, anti-pattern 15, for a detailed description.

24. Sprint Cancellations without Consultation

Observation: The Product Owner deems the Sprint Goal obsolete and cancels the Sprint without consulting the Scrum Team.

Background: Technically, it is the prerogative of the Product Owner to cancel Sprints. However, the Product Owner should not do so without a serious cause. Moreover, the Product Owner should never abort a Sprint without consulting the rest of the Scrum Team. Maybe someone knows how to save the Sprint Goal, provided it is still useful. Misusing the cancellation privilege indicates a serious team collaboration issue and a lack of commitment to living Scrum Values.

Remedy: Take it to the Scrum Team, embrace Scrum Values, and discuss the issue. Generally, Scrum is a remarkably egalitarian affair rooted in collaboration and a lack of authority of the individual. Consequently, it is beneficial for the team's morale and collaboration to include everyone in the decision process. Moreover, the wisdom of the Scrum Team may save the day from time to time.

25. No Sprint Cancellation

Observation: The Product Owner does not cancel a Sprint whose Sprint Goal can no longer be achieved or has become obsolete.

Background: If the Scrum Team identified a unifying Sprint Goal, such as integrating a new payment method, and the management then abandons that functionality mid-Sprint, continuing to work on the Sprint Goal would be wasteful. In such a case of obsolescence, the Product Owner has to consider canceling the Sprint. If the Product Owner fails to do so, continuing the work would create waste while raking up significant opportunity costs. Moreover, nothing is more frustrating than squandering valuable time working on something no one wants. By the way, the fact that the Developers receive payment for the wasted effort does not equalize the issue.

Remedy: Every Scrum Team member should address the issue when it arises immediately. Dare the utmost and talk about it.

> ***Practical Tales:*** *I was working with a startup that was very eager to broaden its customer base by white-labeling its product to other companies. Unfortunately, the application's design did not allow for simply adding a different front end to accommodate the needs of white-labeling customers. Also, there was no chance of customizing the application for that purpose; we had to redo everything manually.*
>
> *So, when the founders pushed for implementing a specific white-label partner, the two Scrum Teams were not excited, but they got to work. Unfortunately, when the project was 90 percent complete, the founders canceled it. It turned out that we started working on the white-label version even before the contract was signed; it never was signed, but no one told us. One of our two best Developers was so mad about this misuse of his time that he quit and became a kindergarten teacher—I am not kidding.*

PRODUCT OWNER ANTI-PATTERNS DURING THE DAILY SCRUM

By comparison to other Scrum events, the Daily Scrum is remarkably resilient to Product Owner anti-patterns.

26. Assignments

Observation: The Product Owner assigns tasks directly to team members during the Daily Scrum.

Background: According to the Scrum Guide, the Developers self-organize their work: "Through discussion with the Product Owner, the Developers select items from the Product Backlog to include in the current Sprint. . . . No one else tells them how to turn Product Backlog items into Increments of value."[7]

In other words, the Developers select the work necessary to accomplish the Sprint Goal and then decide how to build it, meeting the Definition of Done. It is their prerogative as they commit to achieving the Sprint Goal.

Following are some reasons why the Product Owner violates the self-management prerogative of the Developers:

- **Misunderstanding of role:** The Product Owner may not fully understand the Developers' role and self-management principles, leading them to behave more like a traditional project manager, reverting to old habits.
- **Lack of trust:** The Product Owner may lack trust in the Developers' ability to choose and complete tasks effectively.
- **Lack of transparency:** If there's a lack of transparency about progress or quality of work, the Product Owner may step in more directly to ensure things stay on track.
- **Control issues:** The Product Owner might have issues relinquishing control and may trust more in their judgment about task assignments than in the Developers' self-organization abilities.
- **Pressure from higher management:** Higher management may pressure the Product Owner to control and direct work.
- **Time constraints:** An impatient Product Owner may feel that directly assigning tasks saves time and helps to speed up the process.

Remedy: The Product Owner has no authority to assign tasks to Developers. However, merely stating this notion is unlikely to change the situation, particularly when trust issues are at play.

7. Ken Schwaber and Jeff Sutherland, "Sprint Planning," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#sprint-planning.

Instead, consider the following approaches:

- **Role clarification:** Help the Product Owner fully understand their role and responsibilities within Scrum and the importance of Developers' self-management, probably even by formally participating in Scrum training.
- **Building trust:** Facilitate activities that foster trust between the Product Owner and the Developers. Help the Product Owner understand the team's capabilities and strengths.
- **Enhancing transparency:** Invest in Product Backlog refinement and Sprint Backlog visualization as a Scrum Team. Point the Product Owner to the benefits of joining the Daily Scrum to stay up-to-date.
- **Addressing control issues:** Through coaching, help the Product Owner understand that the Developers are professionals who can manage their work effectively, reinforcing the Scrum values of respect, openness, and courage in all team interactions.
- **Management alignment:** Involve higher management in Scrum training and discussions to align their expectations with Scrum principles, mainly the importance of self-management.
- **Effective time management:** Show the Product Owner that letting the team self-manage is not a waste of time but rather an investment in a more effective, high-performing team, as it frees time on their side to better understand customer needs and market expectations. The Product Owner's job is not to micromanage the Developers but to maximize the value of the Scrum Team's work.

27. Introducing Additional Work

Observation: Despite good Product Backlog refinement and Sprint Planning, the Product Owner or even stakeholders attempt to introduce new work to the current Sprint during the Daily Scrum.

Background: Team-internally, this behavior may be acceptable for high-priority bugs, as the Daily Scrum does serve as a prioritization meeting of operational work.

Otherwise, the Sprint Backlog's composition is the Developers' sole responsibility as they commit to accomplishing the Sprint Goal. In return, they earn the privilege of choosing the necessary work to honor their commitment. Consequently, they may accept new work during the Sprint; however, that is their decision. This rule also applies should the Developers deliver the Sprint Goal before the end of the Sprint and now have some spare time. The Scrum Team's productivity focus is

achieving the Sprint Goal, not maximizing utilization rates of the team members or releasing as many Product Backlog items as possible.

The Developers' prerogative of choosing Product Backlog items does not rule out that the Product Owner may pitch additional items to the Developers, though. One of their pitches may be successful.

Another dimension of the anti-pattern is how the Product Owner may have contributed to additional requirements appearing after the Sprint Planning due to inadequate stakeholder management; for example:

- **Lack of stakeholder alignment:** Different stakeholders, who are unaligned on what they want, influence a willing Product Owner, leading to requirements changes.
- **Late feedback:** The Product Owner may receive late feedback from stakeholders, leading them to push new requirements on the team.
- **High market pressure:** The Product Owner may feel pressured to deliver more due to market competition or high stakeholder expectations.

Remedy: There are a few ways to convince the Product Owner to stop pushing the Scrum Team to accept more work during the Sprint.

First, consider the following within the Scrum Team:

- **Clarify roles:** Reiterate the Scrum Guide principles, emphasizing that it's the Developers' role to manage the Sprint Backlog.
- **Highlight Sprint Goal focus:** Emphasize that the main objective of the Sprint is to achieve the Sprint Goal, which the Scrum Team collaboratively identified, not to maximize task completion.
- **Feedback loop:** Encourage open communication and feedback between the Product Owner and the Developers during the next Retrospective to ensure a shared understanding of workloads and expectations.
- **Discuss long-term productivity:** Stress that maintaining a sustainable pace will lead to long-term productivity, whereas overloading may lead to burnout.
- **Promote respect:** Stress the importance of respecting the team's decisions about how much work they can realistically handle.

Second, consider supporting your Product Owner in managing stakeholders and their expectations, for example, by designing and facilitating workshops including

the Product Owner, the Developers, and the stakeholders to create a shared understanding of the Product Goal and how it supports realizing the product strategy. Jeff Patton's user story mapping[8] concept has proven successful in this respect.

SPRINT REVIEW ANTI-PATTERNS

Finally, there is the Sprint Review. Despite that it is an outstanding opportunity for the Product Owner to improve the collaboration with both stakeholders and Developers and figure out collectively in what direction to take the product next, some Product Owners do not get the message.

28. Selfish Product Owner

Observation: Product Owners present the team's accomplishments as their own to the stakeholders.

Background: The Product Owner is not the boss or leader of the Scrum Team. Scrum is a remarkably egalitarian approach: No one has any authority to tell teammates what, when, or how to do something. There are no subroles on Scrum Teams, and there is no hierarchy. Instead, a Scrum Team relies upon self-management and collective responsibility to achieve its goals—the Scrum Team wins together, and the Scrum Team loses together. Consequently, the Product Owner cannot claim the team's success as their own.

Remedy: Remember the old saying: There is no *I* in *team?* I suggest addressing the problem during a Retrospective and initiating a respectful dialogue with the Product Owner to clarify the principles of collective responsibility and success in Scrum.

Other helpful practices to address the Product Owner's behavior include the following:

- Reiterate the five Scrum values—commitment, courage, focus, openness, and respect—emphasizing shared success.

8. User Story Mapping is a concept by Jeff Patton that visually represents the customer journey. It organizes user stories along two dimensions: horizontally, illustrating the sequence of a user's experience, and vertically, denoting priority or complexity. It helps teams understand the system's functionality and its relation to user behavior. It enables them to prioritize work effectively based on value and complexity while including stakeholders in creating the map and a shared understanding. Learn more: Jeff Patton, *User Story Mapping: Discover the Whole Story, Build the Right Product* (O'Reilly, 2014).

- Encourage regular feedback among team members and the Product Owner to foster mutual understanding and respect. Several Liberating Structures micro-structures exist for this purpose, such as Heard, Seen, Respected[9] and the Conversation Café.[10]
- Share stories of successful teams, emphasizing collective effort and achievement.
- Change the format of the Sprint Review; for example, run it as a science fair with individual booths.
- Publicly acknowledge the entire team's contributions when the Scrum Team achieves important goals, such as Sprint Goals or the Product Goal, to reinforce the idea of shared success.

29. "Acceptance" by the Product Owner

Observation: The Product Owner uses the Sprint Review to "accept" Product Backlog items that Developers consider done, as they adhere to the team's Definition of Done.

Background: A formal acceptance of work packages by the Product Owner is not foreseen in Scrum, for there is the Definition of Done. What is done is done, is releasable, and we trust the word of the Developer who turned the Product Backlog item into an Increment. In Scrum, we do not throw anyone under the bus but trust in our teammates.

Remedy: Instead of establishing a formal acceptance, I advise creating feedback loops: Did the Developers deliver the agreed-upon functionality? Also, Product Owners should decouple these feedback loops from the Sprint Review. Instead, they should communicate with the Developers whenever needed or when work items meet the Definition of Done. Typically, Developers will reach out to the Product Owner once they finish a Product Backlog item. There are plenty of opportunities to discuss what the Developers accomplished.

30. The "Know-It-All" Product Owner Loving Their Ideas

Observation: The Product Owner does not accept feedback from stakeholders or the Developers, stubbornly pursuing their own ideas. When presented with

9. Henri Lipmanowicz and Keith McCandless, *The Surprising Powers of Liberating Structures: Simple Rules to Unleash a Culture of Innovation* (Liberating Structures Press, 2013), 244–46.
10. Lipmanowicz and McCandless, *The Surprising Powers*, 224–27.

feedback that does not reinforce their preconceptions, the Product Owner will say that the stakeholders "just don't understand."

See Chapter 8, anti-pattern 6, for a detailed description.

Food for Thought

Product Owner vs. product manager—Do we need a product manager in an organization dedicated to practicing Scrum? Often, particularly in larger organizations, we notice a separation between these two roles: the Product Owner position is a tactical job at the team level, delivering what the product manager created at the organizational level. Is this a beneficial division of responsibilities, given that our Product Owner, according to the process described in Chapter 10, is also responsible for product discovery?

Conclusion

Admittedly, the Product Owner role is the most challenging Scrum role because it requires a balance of very different skills: understanding needs, devising solutions, building consensus, and sometimes managing influential but troublesome stakeholders. A Scrum Team can still achieve great things without great individual Developers, but it will never achieve great things without a great Product Owner.

In this model, the Product Owner serves as the crucial catalyst for value creation as they determine the Scrum Team's next direction. Fortunately, good Product Owners can become great through diligence and practice. These Product Owner anti-patterns provide a starting point.

Toolbox

Regarding the following tips and exercises, you will find detailed descriptions in Appendix B:

- **"20 Questions from a New Scrum Master to Product Owner"** comprises 20 questions addressing the future collaboration between the two individuals and the rest of the Scrum Team.
- **"Agile Metrics—The Good, the Bad, and the Ugly"** points to suitable agile metrics reflecting either a Scrum Team's progress in becoming agile or an organization's progress in becoming a learning organization.
- **"Building Communities of Practice"** helps win hearts and minds within the organization. It provides authenticity to the agile transformation—signaling that the effort is not merely another management fad.
- **"Creating Product Goals as a Collaborative Practice"** suggests using Ralph Jocham's Product Goal Canvas to create a shared understanding among the Scrum Team members about the why, what, and how of the team's journey.
- **"Stakeholder Communication Tactics"** comprise eleven proven stakeholder communications tactics to help you not just do Agile but *become* agile.

This page intentionally left blank

3 Scrum Developer Anti-Patterns

Introduction

This chapter addresses Developer anti-patterns, from corner cutting to meet deadlines to entertaining side gigs to delivering undone work. Learn more about what to look out for if you want to support your teammates who build the Increment; see Figure 3.1.

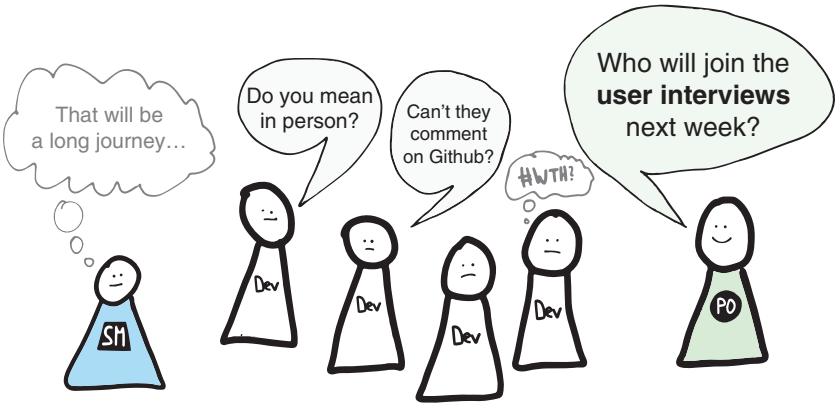

Figure 3.1 Product development is a team sport in real life: routinely, we all have close contact with our customers, not just at Sprint Reviews.

THE ROLE OF THE DEVELOPERS IN SCRUM

The term *Developer* does not limit the role to technical people, such as software engineers. Instead, in the spirit of the Scrum Guide, the role of Developer is much more inclusive: everyone who contributes to the creation of the Increment is a Developer—lawyers and marketers included.[1]

Developers are accountable for the following:

- Creating a Sprint Goal collectively with the Product Owner and Scrum Master[2]
- Committing to the Sprint Goal
- Picking all Product Backlog items they deem necessary to accomplish the Sprint Goal, transferring them to the Sprint Backlog
- Alternatively, creating new Product Backlog items and adding them to the Sprint Backlog
- Creating a plan for the Sprint
- Adapting their plan toward achieving the Sprint Goal during the Daily Scrum
- Adhering to a Definition of Done to ensure quality
- Holding each other accountable as professionals

They enjoy complete autonomy in planning and executing their work to transform Product Backlog items into Increments of value. This approach aligns with the Agile Manifesto's emphasis on self-organizing teams and valuing individuals and interactions over processes and tools.

So, let's dive into an array of common Scrum Developer anti-patterns that signal to Scrum Masters that their teammates need support.

DEVELOPER ANTI-PATTERNS BY SCRUM EVENTS

SPRINT ANTI-PATTERNS

Most of the following Developer anti-patterns result from a lack of focus or procrastination.

1. Ken Schwaber and Jeff Sutherland, "Developers," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#developers.
2. Ken Schwaber and Jeff Sutherland, "Sprint Planning," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#sprint-planning.

1. No WiP Limit

Observation: The Developers do not choose to employ a work-in-progress (WiP) limit to streamline the flow of work.

Background: The purpose of the Sprint is to deliver to customers a potentially valuable product Increment that also supports the organization's sustainability. Achieving this goal requires focus to accomplish the necessary work to meet the Sprint Goal by the end of the Sprint.

Flow theory suggests that a team's productivity improves with a WiP limit.[3] The WiP limit defines the maximum amount of work that the Developers can undertake simultaneously. Exceeding the WiP limit by pulling in more work than the available capacity can handle will reduce the overall throughput of the Developers. The cycle time—the period between starting and finishing a ticket—measures this effect.

Despite the apparent benefits of minimizing WiP (i.e., starting work on a work item only when the team has capacity *to* work on it), Developers may believe that increasing WiP is a good thing for the following reasons:

- **They misunderstand productivity:** Developers may equate busyness with productivity, mistakenly believing that working on multiple tasks simultaneously leads to greater output.
- **They are afraid of idle time:** Developers may fear appearing unoccupied, leading to a compulsion to fill their capacity.
- **They are under external pressure:** Stakeholders' or managers' demands may pressure Developers into taking on more tasks than they can effectively handle.
- **They don't understand the costs of WiP:** Developers may not understand the benefits of limiting WiP and its positive impact on cycle time and sustainable work speed.

Remedy: There are several options to help the Developers to embrace the idea that less is more. Here are ways to introduce a WiP limit:

- **Education:** Organize flow theory and WiP learning sessions. Explaining the concept of context switching and how it affects productivity can be particularly effective.

3. Wikipedia, s.v. "Kanban (Development)," updated August 17, 2023, https://en.wikipedia.org/wiki/Kanban_(development)#Managing_workflow.

- **Metrics:** Suggest metrics such as cycle time or throughput to measure, and track the effect of WiP limits on team performance, providing hard data on the benefits.
- **Experimentation:** Encourage the team to experiment with WiP limits. Start with a higher limit and gradually decrease it. Let the team experience first hand how it affects their productivity and stress levels.
- **Visualization:** Use Sprint boards to visualize the amount of WiP. This visual can help the Developers understand how much work is in progress and identify bottlenecks.

2. Lack of Professionalism

Observation: The Developers selectively pull work from the Sprint Backlog based on their personal interests, not on what the Scrum Team needs to get done next. They focus on the creative part and procrastinate in accomplishing the more tedious parts of the work. Consequently, you notice tickets queueing in the code-review column.

Background: This selection effect often overlays with the missing WiP limit issue. Human beings are motivated by short-term gratifications. Coding a new task off the board feels like successfully piecing together another part of the puzzle.

By comparison to this dopamine fix,[4] checking how someone else solved another problem during code review is less rewarding. The same applies to tedious or complex tasks that may seem daunting or boring. Moreover, Developers may select tasks aligned with their expertise or interests, leaving other essential yet less appealing tasks undone.

The behavior described suggests that the Scrum Team is still in the early stages of team maturity. They lack a shared commitment to tasks; there is an imbalance in the distribution of work and a tendency to gravitate toward individually appealing rather than collectively beneficial tasks.

Conversely, mature Developers have a shared understanding of the team goals and prioritize work that benefits the team's objectives over individual interests. They understand the importance of finishing tasks and adhere to the agreed-upon Definition of Done.

4. Marc Dingman, "2-Minute Neuroscience: Reward System," Neuroscientifically Challenged, February 13, 2015, YouTube video, 2:03, https://youtu.be/f7E0mTJQ2KM.

Remedy: What course of action should you take to help the Developers overcome this anti-pattern? There are a few possibilities:

- **Facilitate collaborative decision-making:** Encourage Developers to discuss and decide which tasks to work on during Sprint Planning. This collective decision-making can foster a sense of shared responsibility and discourage cherry-picking of tasks.
- **Promote teamwork:** Encourage pair programming or mob programming, where multiple Developers work together on the same task. This practice can promote collaboration and shared learning.
- **Reward completion over initiation:** During the Daily Scrum, celebrate the completion of tasks more than initiating new ones. "Walking the Sprint board"[5] supports this approach.
- **Foster a growth mindset:** Encourage Developers to view challenging or tedious tasks as opportunities to learn and grow rather than as burdens.
- **Adopt a pull system:** Suggest implementing a Kanban-style pull system where Developers can start new work only when they have completed their previous task.

3. Board Out of Date

Observation: The Developers do not update tickets on the Sprint board in time to reflect current statuses, visualizing progress toward the Sprint Goal.

Background: Even though the Scrum Guide does not mention them, Sprint boards can help a Scrum Team by providing transparency that helps them forecast progress and coordinate their work. Whether it is a physical or digital board, it is also an integral part of the communication of the Scrum Team with its stakeholders.

A Sprint board that is not up-to-date may cause stakeholders' to lose trust in the Scrum Team and impose more traditional project management methods like PRINCE2.[6]

5. "Walking the board" is a practice to review and visualize a team's progress toward the Sprint Goal. It involves a reverse review of tasks on the board, starting from the Done column and moving toward To Do. The Scrum Team discusses each task's status, identifies obstacles, and seeks improvements. This regular and collaborative Daily Scrum practice emphasizes completion, transparency, and continuous process enhancement, fostering a collaborative and iterative approach to work.
6. Wikipedia, "PRINCE2," updated October 3, 2023, https://en.wikipedia.org/wiki/PRINCE2.

Remedy: As the Developers commit to accomplishing the Sprint Goal and self-manage their work to meet their commitment, they decide how to keep track of their progress during the Sprint. A Sprint board is only one option but a popular one.

It is not the task of the Scrum Master to maintain a Sprint board, for example, by moving tickets. The Developers should do this during the Daily Scrum if they consider using a Sprint board helpful. It is also not the responsibility of the Scrum Master to update an online board so that it reflects the statuses of a corresponding physical board.

Developers can decide not to employ a Sprint board, but if they choose to use a Sprint board, they must also commit to updating the board regularly.

4. Side Gigs

Observation: The Developers worked on issues outside the Sprint Goal, and the Product Owner learned about those issues for the first time during the Sprint Review.

Background: This Sprint Review anti-pattern is truly "rocking the boat." Focus, commitment, and openness—just to name the most apparent Scrum values violated here—are the first principles for collaboration among Scrum Team members.

Anything foreseeable that the Developers want to address during the Sprint needs to be communicated during the Sprint Planning to ensure transparency, which in turn allows for the creation of a reasonable, achievable Sprint Goal. This Sprint Goal and its accomplishment is critical to build trust with stakeholders, thus supporting the long-term team development and value creation. Running a shadow operation by syphoning off capacity to other tasks undermines that effort.

However, there might be an exception that proves this rule: Suppose the Scrum Team's Product Owner has a dominant personality and relentlessly pushes to ship as many new features as possible. Moreover, the management supports the Product Owners in their approach, as they consider this an excellent way to maximize output. Consequently, there is little or no time to attend to refactoring or bug fixing unless the Developers tackle these jobs under the radar. In this situation, I would have sympathy for the approach as a stop-gap measure before solving the underlying challenge of the pushy Product Owner.

Remedy: I recommend addressing the issue head-on by exploring the reasons for the clandestine operation during a Retrospective:

- **Why** is there a lack of trust on the side of the Developers that motivated them to muddy transparency and violate Scrum values?
- **What** work was so important that the Developers decided to keep it confidential? Is that work critical to accomplish the current Product Goal? If not, why did they choose to work on it anyway?
- **Who** commissioned the work? Is a stakeholder possibly bypassing the Product Owner by directly assigning work to Developers?

The Scrum Team should also reconsider their allocation of their capacity, perhaps setting aside 15 to 20 percent of their total capacity to work on technical issues such as refactoring or bug fixing, at the discretion of the Developers.

5. Gold-Plating

Observation: The Developers increase the scope of the Sprint beyond the Sprint Goal by adding unnecessary work to the Product Backlog items of the Sprint Backlog; see Figure 3.2.

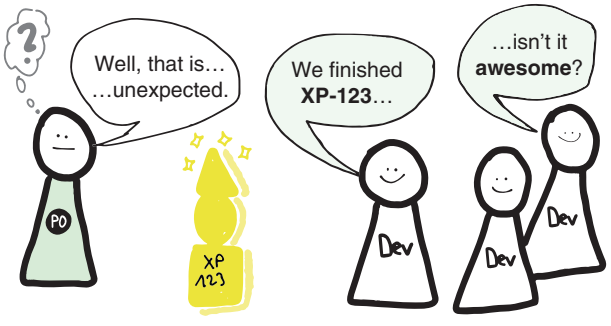

Figure 3.2 If Developers ignore the original scope agreement with the Product Owner, it results in gold-plating, diminishing the value created.

Background: We call this issue scope-stretching or gold-plating. The Developers ignore the original scope agreement with the Product Owner. For whatever reason, the team enlarges the task without first consulting the Product Owner. This ignorance may result in a questionable allocation of development time.

Remedy: There is a simple solution: the Developers and the Product Owner need to talk more often, creating a shared understanding from product vision down to the individual Product Backlog item, thus improving the trust level. If the Product Owner is not yet co-located with the Developers, now would be the right moment to reconsider. Alternatively, a distributed Scrum Team may invest more time in how best to communicate synchronously and asynchronously to improve its level of collaboration and alignment.

6. Cutting Corners to Meet an Increment's Arbitrary Deadline

Observation: The Developers reduce product quality below the level defined in the Definition of Done to meet an imposed release deadline.

Background: The Definition of Done serves a critical role in the Scrum framework: it represents the quality standard that signals to everyone what the Developers need to accomplish to create a potentially releasable Increment. By *releasable,* we mean we can deliver the Increment to our customers and users without legal, financial, or ethical repercussions.

Watering this quality level down to meet a—probably arbitrary—deadline imposed on the team puts the team's success at risk because it introduces undone work and technical debt. Both of these consequences of neglecting the quality standard will impede the Scrum Team's ability to be innovative at a later stage.

Consequently, it is crucial for a Scrum Team's success that you challenge processes in the organization that result in arbitrary deadlines. While there are valid reasons for deadlines—for example, a general legal requirement is imposed—we are interested in the less transparent reasons, such as the following:

- Have salespeople sold future features without involving the Scrum Team in the decision? Are those features already legally enforceable based on a contract? I would consider that an unfriendly act, as the sales department is aiming for a local optimum: meeting a sales quota at the expense of the Scrum Team.
- An alternative scenario would be when stakeholders consider imposing deadlines as a means of last resort, as the Scrum Team is notoriously unreliable about forecasting the availability of critical features. Here, the use of a deadline is a sign of—probably justified—mistrust rather than a sinister move on behalf of the stakeholders.

Remedy: There are several ways a Scrum Team can address the arbitrary deadline issues:

- In the case of the sales team, the Scrum Team would be well-advised to understand the process and its motivation and seek allies among other stakeholders and the management to work on overcoming this impediment in the future. There is no quick fix, as this kind of behavior of optimizing for local optima is usually deeply rooted in the organization's culture, reflected, for example, in the rewards and compensation system.
- Also, double down on getting buy-in from stakeholders regarding your Definition of Done.
- Generally, it is a good idea for the Scrum Team to embark on a journey of trust-building among stakeholders. The best way to achieve that is to deliver valuable Increments regularly. Doing so encourages stakeholders to reach out to the team and ask for its support to solve their challenges instead of bypassing the team in the decision process.

It is paramount for a Scrum Team's success that stakeholders understand that there is no business agility without technical excellence. The latter starts with adhering to the Definition of Done at all times.

Moreover, constantly meeting an arbitrary deadline is not sustainable at the team level. This culture quickly leads to burnout.

7. What Technical Debt?

Observation: The Product Backlog does not allocate sufficient capacity to maintain the platform, as the Developers do not demand adequate time to tackle technical debt[7] and bugs and preserve the quality standard of the product. Instead, they fully embrace the feature factory,[8] shipping one new feature after another.

7. Technical debt is a metaphor in software development describing the future work and complexity resulting from short-term solutions over better, long-term alternatives. Accumulating over time, it can lead to increased maintenance costs, slower development, and lower software quality. Learn more: Martin Fowler, "Technical Debt," May 21, 2019, https://martinfowler.com/bliki/TechnicalDebt.html.
8. John Cutler's feature factory concept describes a product development environment where teams focus on delivering features rather than creating value for users and the business. In a feature factory, output (the number of features) precedes outcomes (user experience, the desired change in behavior, accomplishing business objectives, etc.). Unfortunately, this misaligned focus can lead to wasted effort and resources. Scrum practices combat the feature factory by emphasizing collaboration, iterative development, and continuous feedback. Cutler introduced this concept in his article "12 Signs You're Working in a Feature Factory," @johncutlefish's blog, November 17, 2016, https://cutle.fish/blog/12-signs-youre-working-in-a-feature-factory.

See Chapter 10, anti-pattern 18, for a detailed description.

8. No Slack Time

Observation: The Developers do not demand 20 percent slack time—or unplanned capacity—from the Product Owner; see Figure 3.3.

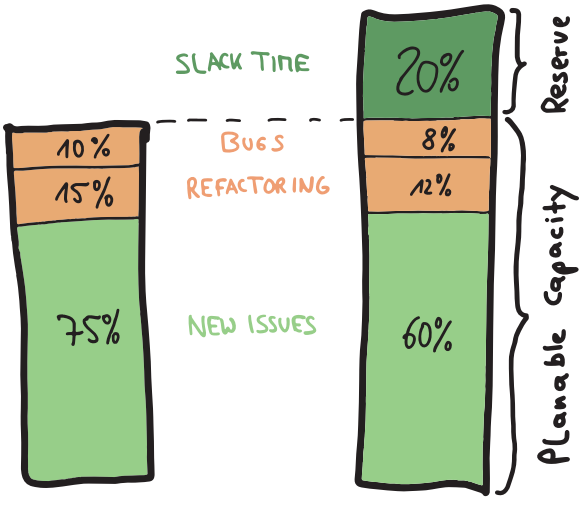

Figure 3.3 Capacity planning in complex environments requires considering the unpredictable and embracing the idea of slack time.

Background: Making slack time a part of the Scrum Team's work has many long-term advantages, including the following:

- **It promotes innovation:** By leaving room for unplanned capacity, Developers can explore new ideas and technologies, boosting productivity and/or quality.
- **It supports sustainable pace:** Avoiding 100 percent utilization helps to prevent burnout and promotes a sustainable pace of work, maintaining team morale and productivity over time.
- **It improves quality:** Developers can devote time to improving code quality, refactoring, or writing better tests, leading to fewer bugs and less technical debt.
- **It supports team building:** It gives space for activities that strengthen team relationships and improve communication, leading to better collaboration, for example, by pairing with each other to share knowledge and skills.

- **It allows for unexpected issues:** Unplanned capacity can absorb unforeseen issues that could disrupt Sprint's progress.

Conversely, if an organization insists that a Scrum Team is always fully (100 percent) utilized, whether by dictate or incentives, the team performance will usually suffer as a result.

Remedy: Despite the advantages of slack time or unplanned capacity, this way of working is a hard sell, as many organizations still think in terms of the industrial age, where maximizing utilization is a highly praised achievement.

Changing this bias requires patience and perseverance. Approaches that may help include the following:

- **Running a pilot project:** Suggest running a controlled experiment with one Scrum Team first. This way, management can see the results of implementing slack time without committing all teams.
- **Showing trust in the team:** Advocate that giving the team autonomy to manage their work shows trust, which can improve morale and productivity.
- **Increasing transparency:** Commit to regularly sharing progress and outcomes during Sprint Review, so management can see how slack time positively affects the team's productivity and the product's quality.
- **Demonstrating the long-term benefits:** Explain how slack time could lead to long-term benefits, such as reducing burnout, increasing innovation, decreasing technical debt, and improving team morale.
- **Highlighting competitive advantage:** Argue that allowing slack time can be a competitive advantage, enabling faster learning and adaptation in a fast-changing business environment.

In addition, Scrum delivers another perspective on the benefits of slack time:

- **Empiricism:** Scrum is based on empirical process control, meaning that knowledge comes from experience and decision-making based on observed reality. Slack time allows for this learning and adaptation to take place.
- **Self-management:** Scrum emphasizes the importance of self-managing teams. Allowing slack time is a trust exercise that reinforces a Scrum Team's autonomy and encourages self-management.

- **Focus:** During a Sprint, Scrum Teams focus on the Sprint Goal. Slack time provides the space for teams to concentrate intensely on their work.
- **Commitment:** Scrum Teams commit to achieving the goals of the Sprint. When management allows slack time, they generally show their commitment to the team and Scrum.
- **Courage:** It takes courage from all sides—Scrum Team and management—to challenge traditional working norms and allow slack time. But this courage can improve team morale, productivity, and product quality.

It may take time to convince your management and Product Owner to accept slack time. However, if you start such an initiative, prepare yourself for the inevitable; slack time is not your typical Retrospective action item but attempts to change work patterns deeply rooted in an organization's tradition.

SPRINT PLANNING ANTI-PATTERNS OF THE DEVELOPERS

Scrum's Sprint Planning aims to align the Developers and the Product Owner on what to build next, delivering the highest possible value to customers:

9. No Capacity Check

Observation: The Developers overestimate their available capacity, resulting in a too-ambitious Sprint Goal.

See Chapter 6, anti-pattern 1, for a detailed description.

10. Planning Too Detailed

Observation: During the Sprint Planning, the Developers plan every single task of the upcoming Sprint.

See Chapter 11, anti-pattern 5, for a detailed description.

11. Too Little Planning

Observation: The Developers skip planning altogether and start working immediately.

Background: If a Scrum Team is proficient in Product Backlog refinement, maintains an actionable Product Backlog, and the team already anticipated the new

Sprint Goal, they need little time to create a Sprint Plan. Even for these teams, skipping planning is a mistake, as it not only addresses who will work on what, it is also an excellent opportunity to talk about various aspects of the upcoming work, improving product quality, or sharing skills. Planning among Developers addresses, for example,

- How to spread knowledge among the Developers, such as by pairing or mobbing,
- Where the architecture is heading, and
- How to reduce technical debt.

Remedy: Scrum Masters should work to convince planning-resistant Developers that creating a Sprint Plan is beneficial for the success of the Scrum Team. Compelling reasons include the following:

- **The Scrum Team needs a shared vision:** Creating a Sprint plan ensures everyone has a shared understanding of the Sprint Goal and the steps required to achieve it.
- **The Scrum Team needs to mitigate risk:** The planning helps identify and address potential risks or blockers early rather than dealing with them reactively when they become an issue.
- **Having a Sprint Plan does not impair flexibility:** Scrum is not about sticking rigidly to the plan but provides a baseline from which to adapt as new information comes to light. There is even a Scrum event, the Daily Scrum, to inspect progress toward the Sprint Goal and adapt the plan if necessary.
- **Collaboration during Sprint Planning improves the Scrum Team:** Collective planning fosters collaboration and team cohesion, improving problem-solving and productivity.
- **Scrum Team members need to be accountable for their work:** A Sprint plan holds each team member accountable for their tasks, creating a sense of responsibility and dedication toward meeting the Sprint Goal.

If your Scrum Team hasn't taken planning seriously, get them to try Sprint Planning as an experiment during the next Sprint. Use the Sprint Retrospective to assess whether it improved the Scrum Team's effectiveness to determine if they should continue using the Sprint Planning event.

12. Team Leads?

Observation: The Developers do not devise a plan to deliver their forecast collaboratively. Instead, a *team lead* does all the Sprint planning, and may even assign tasks to individual Developers.

See Chapter 6, anti-pattern 7, for a detailed description.

ANTI-PATTERNS DURING THE DAILY SCRUM

The Daily Scrum is a critical event for inspection and adaptation, run by the Developers, and for guiding the team for the next 24 hours to achieve the Sprint Goal. The Daily Scrum is the shortest planning horizon in Scrum and thus is highly effective in contributing to the Scrum Team's success.

Contrary to popular belief, its 15-minute timebox is not intended to solve all the issues addressed during the Daily Scrum. It is about creating transparency, thus triggering the inspection. If an adaptation of the plan or the Sprint Backlog, for example, is required, the Developers are free to handle the resulting issues at any time. In my experience, most Daily Scrum anti-patterns result from a misunderstanding of this core principle.

13. No Routine

Observation: The Daily Scrum does not happen at the same time and in the same place every day, or even at all.

See Chapter 7, anti-pattern 3, for a detailed description.

14. The Lost Scrum Team

Observation: The Developers don't know whether they make progress toward accomplishing the Sprint Goal.

Background: The Daily Scrum serves one purpose as it answers two simple questions: Are we on track to meet the Sprint Goal? And, if not, do we need to adapt the Sprint Plan, the Sprint Backlog, or both?

There are a number of reasons why the Developers cannot tell whether they are progressing toward the Sprint Goal, including the following:

- **Unclear Sprint Goal:** The Sprint Goal may be opaque or overly complex.
- **Unstructured tasks:** The Developers may have failed to refine Product Backlog items adequately, leading to confusion about what they need to accomplish and how to achieve it.
- **Scope creep:** Frequent changes to the Sprint Backlog during the Sprint can lead to a sense of disorientation.
- **Lack of skills or resources:** The Developers don't have the necessary skills or resources to complete their tasks.
- **Insufficient feedback:** The Product Owner may fail to provide feedback to the Developers.
- **Absence of metrics:** The Developers apply no clear metrics or indicators to track progress.

Remedy: Visualizing the progress toward the Sprint Goal, for example, using a Kanban board, is useful. Visualization won't solve the more critical issues mentioned here, but it can signal that the team needs to employ countermeasures.

Regarding this anti-pattern, a Daily Scrum session based on walking the Kanban[9] board—what can the Scrum Team do to move the items closest to Done to Done[10]—works probably better than a usual round-robin-style session, where everyone updates the rest of the team on where they are and what they plan to do.

15. Unpreparedness

Observation: Developers come unprepared to the Daily Scrum.

Background: Developers are often busy and can get lost in their work. They may see the Daily Scrum as an unwanted interruption of more important work and overlook

9. Kanban is a lean practice designed to manage and improve work across systems. It visualizes work and workflow to identify bottlenecks and promote incremental, evolutionary change. Fundamental principles include limiting WiP, making work policies explicit, and using feedback loops. Learn more about Kanban with David J. Anderson's book *Kanban: Successful Evolutionary Change for Your Technology Business* (Blue Hole Press, 2010).
10. Gary Straughan, "Agile Daily Standup—How to Walk the Board (aka Walk the Wall)," Development That Pays, November 4, 2015, YouTube video, 5:23, https://youtu.be/316qdj10j9M.

the reality that everyone needs help from their peers and fellow team members from time to time.

Taking a a brief amount of time to prepare for the Daily Scrum, and then actively participating, is a small price to pay for the benefits of helping move the whole team forward toward its goal and for getting help when one needs it.

Remedy: If Developers truly have better things to do, things that are more important to the team's progress than attending the Daily Scrum, they should do those things. But how do they know those things are more valuable than helping a fellow team member resolve something that may, if not handled, prevent the entire team from achieving its Sprint Goal?

At some point, everyone needs help. Developers should think of attending the Daily Scrum as providing opportunities to pay the team back for help they have given or paying it forward for help they will receive from fellow team members in the future. A few minutes of preparation and a 15-minute (maximum) meeting is a small investment with potentially great returns.

So as not to interrupt Developers in the midst of concentrated work, many Scrum Teams schedule their Daily Scrum meetings first thing in the morning or just before or after lunch. Some Developers who come in early or who are skipping lunch may be interrupted, but a mid-morning or mid-afternoon meeting is sure to interrupt almost everyone.

16. The Daily Status Report

Observation: The Daily Scrum is a daily status report meeting, and Developers take turns to "report" progress to the Scrum Master, the Product Owner, or maybe even a stakeholder.

Background: Unfortunately, the old "three Daily Scrum questions" often serve as a template for this anti-pattern:

- Answering them is simple and efficient, and
- It creates a comforting routine.

However, focusing on what the Scrum Team did yesterday, what it is working on today, or what it is not working on because it is blocked can obscure the more important question: Will the Scrum Team meet the Sprint Goal?

Remedy: Break the reporting cycle by abandoning the old Daily Scrum questions template. The aforementioned walking the Kanban board approach is one way to do this.

In addition, the following changes may support breaking out of the reporting mode:

- **Rotate facilitator role:** Encourage each team member to take turns facilitating the Daily Scrum, as different perspectives may lead to new and creative approaches.
- **Incorporate feedback:** Seek feedback from Developers about the Daily Scrum's effectiveness, for example, by running anonymous surveys. Then, use this inspection to adapt the format frequently; waiting for the next Retrospective is unnecessary.
- **Prepare for the Daily Scrum:** Encourage team members to prepare for the Daily Scrum by reviewing their progress and upcoming tasks, individually or collaborating with teammates.

17. Planning Meeting

Observation: The Developers hijack the Daily Scrum to discuss new requirements, refine user stories, or have a planning meeting, probably collaborating with the Product Owner.

See Chapter 7, anti-pattern 8, for a detailed description.

18. Problem-Solving Session

Observation: During the Daily Scrum, the Developers trigger discussions to solve problems instead of parking the problems to be addressed after the Daily Scrum.

Background: The Daily Scrum is not designed to be a problem-solving session; it's a progress-sharing and problem-identification session. When Developers discover problems that need to be solved or issues that need to be addressed, they do so outside the Daily Scrum.

This strict focus may appear dogmatic. However, there are good reasons for it:

- **Promotes autonomy:** Dealing with problems outside the Daily Scrum promotes the Scrum Team's self-management and problem-solving capabilities.
- **Respects time:** Keeping problem-solving separate from surfacing issues respects the time of team members who may not be involved in resolving a specific problem.

- **Ensures thorough problem-solving:** Problems may require in-depth discussion, brainstorming, and the involvement of additional stakeholders. These activities can be more effectively accomplished outside the Daily Scrum, avoiding rushing a solution.
- **Prevents unnecessary discussion:** The short timeframe prevents unproductive or premature discussions, keeping the Daily Scrum focused and actionable.
- **Preserves energy:** The 15-minute timebox keeps the team's energy. Problem-solving can be time-consuming and mentally taxing during the Daily Scrum, particularly if the Scrum Team members haven't prepared themselves for it.

Remedy: Besides educating everyone on the purpose of the Daily Scrum, there are a few other things you can try:

- **Walk the board:** When using a Kanban-like Sprint board, focus the Daily Scrum by starting the inspection with WiP items closest to being Done. Then move column by column to the left and mark issues for the follow-up problem-solving session.
- **Create a parking lot:** Alternatively, introduce a parking lot concept for issues that emerge during the Daily Scrum. List these issues separately, and deal with them in a dedicated problem-solving meeting after the Daily Scrum.
- **Establish problem ownership:** If the Scrum Team identifies a problem, collectively look for a team member to take ownership for organizing a separate discussion to solve it. Move to the next issue only when this team member has acknowledged their responsibility.
- **Encourage feedback:** Seek feedback from the team about the structure and effectiveness of the Daily Scrum, and use their suggestions to improve the meeting. They may be more supportive if they understand the rationale behind not including problem-solving in the Daily Scrum.

If everything fails, run an experiment: choose a Sprint to lift the timebox on the Daily Scrum, and dedicate that Sprint's Retrospective to analyzing the outcome.

19. Excessive Feedback

Observation: Team members criticize other team members immediately after their contribution, sparking an argument instead of taking their critique outside the Daily Scrum.

See Chapter 7, anti-pattern 5, for a detailed description.

20. Monologuing

Observation: Team members use the Daily Scrum to monolog about topics outside the scope of the Daily Scrum.

See Chapter 7, anti-pattern 10, for a detailed description.

21. Not Supporting Struggling Developers

Observation: A Developer experiences difficulties accomplishing an issue over several consecutive days, and nobody on the Scrum Team offers help. Moreover, the Scrum Master fails to facilitate the necessary discussion.

See Chapter 1, anti-pattern 12, for a detailed description.

22. No Impediments

Observation: No Developer reports any obstacles; no one needs help or support from their teammates.

Background: In a complex environment, the lack of any issues is suspicious. If concerns are not discussed during the Daily Scrum, the Scrum Team needs to understand the cause of the problem: without trust, there will be no transparency. Without transparency, there will be no inspection of the progress and no plan adaptation, probably resulting in a failure to meet the Sprint Goal.

Consequently, the Scrum Team loses the benefit of collective intelligence and problem-solving, at the same time undermining the team's efforts to continuously improve its way of working.

Consider some of the following reasons why no impediments are being reported:

- **Previous negative experiences:** Unpleasant past experiences in reporting an obstacle or seeking help can discourage Developers from doing so in the future.
- **Lack of trust or psychological safety:** Developers may feel uncomfortable sharing issues due to a lack of trust within the team.
- **Fear of being judged:** Developers may fear being judged or criticized for facing obstacles and not getting ahead with their work.
- **Perceived weakness:** Some Developers may view asking for help as a sign of weakness or incompetence and prefer to solve problems independently, appearing competent to their peers and superiors.

- **Introverted members:** Introverted Developers may withhold concerns in meetings to avoid unproductive discussions. Instead, they often prefer direct problem-solving with trusted team members outside formal gatherings, which shows the need for diverse communication channels.
- **Shy members:** From the perspective of a shy Developer, not voicing concerns during the Daily Scrum may stem from feeling a lack of psychological safety and respect within the team, suggesting a deeper issue of team dynamics.

Remedy: How can the communication and trust challenges within the Scrum Team best be resolved?

All of the preceding points indicate that the team's psychological safety may be compromised. Perhaps the organization's culture is generally not supportive of openly sharing problems and failures and asking for help. The role of the Scrum Master in this situation is crucial in fostering a more open, trusting, and supportive environment.

Here are some ways to address the issue:

- **Normalize asking for help:** Reinforce the idea that everyone sometimes needs help by demonstrating vulnerability yourself and sharing your challenges and how you've sought help. You can do this at different levels during the Daily Scrum or Retrospective.
- **Promote positive feedback:** Encourage team members to share positive feedback and appreciate each other's willingness to share problems.
- **Promote a no-blame culture:** Foster an environment where the Scrum Team views mistakes as learning opportunities, not failures. Encourage team members to see them as a chance for improvement, not a reason for criticism.
- **Build trust:** Facilitate team-building activities that foster trust and understanding within the team. Encourage open and transparent communication as outlined in the Scrum Values.
- **Address negative experiences:** During the next Retrospective, address past negative experiences that affect Developers' willingness to share.
- **Individual coaching:** Offer one-on-one coaching to Developers who struggle to share their obstacles, providing a safe space for them to express their concerns.

Developer Anti-Patterns Concerning the Sprint Review

Are we still on track to accomplish the Product Goal? Answering this question and adapting the Product Backlog in a collaborative effort of the Scrum Team with internal and external stakeholders is the purpose of the Sprint Review. Some common anti-patterns are listed next.

23. Death by PowerPoint

Observation: The Developer bores participants of the Sprint Review to death with PowerPoint presentations.

See Chapter 8, anti-pattern 7, for a detailed description.

24. Same Faces Again

Observation: It is always the same Developers who participate in the Sprint Review; however, some Developers never participate.

See Chapter 8, anti-pattern 8, for a detailed description.

25. Showing Off Undone Work

Observation: More often than not, the Developers also show work items that are not done.

Background: The Scrum Guide has a simple rule for Sprint Reviews: Never show work that is not done and thus releasable, as this work does not contribute to accomplishing Sprint Goals. Consequently, if the Developers regularly show undone work to Sprint Review attendees, it violates the concept of Done, one of Scrum's first principles, and disrespects the attendees' time commitment.

However, there may be good reasons to show unfinished work on some occasions; for example:

- **Transparency:** Showing undone work can be a way to communicate transparently about the team's progress, particularly if the work is complex or has faced significant challenges.
- **Feedback:** If complex undone work is nearing completion, Developers might benefit from early feedback from stakeholders to steer the work in the right direction.

- **Understanding challenges:** Presenting incomplete work can help stakeholders understand the difficulties encountered during the Sprint and why the Scrum Team failed to complete other Product Backlog items from the Sprint Backlog.
- **Setting expectations:** It can help manage stakeholder expectations for the next Sprint by giving them an idea of what is in the pipeline. This openness may benefit Product Backlog items that cannot be accomplished within a single Sprint for technical reasons.
- **Trust building:** Openly discussing incomplete work can build trust with stakeholders, demonstrating the team's commitment to quality over speed.
- **Collaboration:** It could invite suggestions or assistance from stakeholders with expertise or resources to aid completion.

Remedy: There is no need for a successful Scrum Team to demonstrate to stakeholders that they are worth their paychecks. There is a good reason that the Scrum Guide excludes undone work from the Sprint Review.

However, it may be beneficial to talk about undone in advance occasionally; don't make it a habit, though.

SPRINT RETROSPECTIVE ANTI-PATTERNS OF DEVELOPERS

Sprint Retrospective? I assume all peers agree that even the simplest form of a Retrospective—if held regularly—is far more helpful than having a fancy one once in a while, not to mention having none. This section presents some notorious anti-patterns from the Developers' side.

26. Dispensable Buffer

Observation: The Scrum Team cancels Retrospectives if more time is needed to accomplish the Sprint Goal.

Background: Treating the Sprint Retrospective as an optional event that provides usable slack in the Sprint schedule is a clear sign that the team sees no value in the Sprint Retrospective. I believe it is even a worse anti-pattern than #NoRetro because canceling a Retrospective to achieve a Sprint Goal points to more critical issues.

Remedy: When the Scrum Team uses the Retrospective as a dispensable buffer, they demonstrate that they do not understand the foundational principles of Scrum,

such as empiricism and continuous improvement: in a typical Sprint length of one week or more, skipping a 60- to 120-minute event does not significantly change the time the team has available to progress toward their Sprint Goal. When a Scrum Team consistently fails to meet the Sprint Goal, they need to reflect on why and decide what they should do about it. Guess which Scrum event is designed for that purpose?

As you cannot force anyone to participate in a Retrospective, ask your teammates for their support by skipping the idea of skipping the Retrospective. Then, I would attempt to get the team back on track by focusing strictly on the root cause of the failure to accomplish the Sprint Goal and take it from there. Help your team members to recognize that they can utilize self-management to improve their working conditions.

27. Not Improving the Definition of Done

Observation: The Scrum Team does not regularly inspect the Definition of Done and discuss whether it needs to improve.

See Chapter 15, anti-pattern 12, for a detailed description.

28. Extensive Complaining

Observation: The team members use the Retrospective primarily to complain about the situation and assume the victim's role without taking action to change.

See Chapter 9, anti-pattern 11, for a detailed description.

ANTI-PATTERNS AT THE PRODUCT BACKLOG LEVEL

Scrum is a tactical framework to build products, provided you identify what is worth making in advance. But even after a successful product discovery phase, you may struggle to create the right thing in the right way if your Product Backlog is not up to the job—garbage in, garbage out. Some forms by which Developers contribute to mediocre team success are listed in this section.

29. No Time for Refinement

Observation: The Developers do not have enough Product Backlog refinement sessions, resulting in a low-quality backlog.

Background: The Scrum Guide no longer advises spending up to 10 percent of the Developers' time on the Product Backlog refinement. However, investing in refinement is still a sound business decision: nothing is more expensive than a feature that is not delivering any value.

There may be several reasons for this anti-pattern. For example:

- Stakeholders may pressure the Scrum Team to meet tight deadlines or handle multiple projects, leaving the Developers little time for refinement sessions.
- Alternatively, a lack of team members can lead to Developers being stretched thin and unable to dedicate time to Product Backlog refinement.
- The Developers may not recognize the importance of refinement sessions. For example, poorly structured or unproductive refinement meetings in the past may discourage Developers from participating in future sessions.
- If the Developers' roles and responsibilities in Product Backlog refinement sessions are unclear, they may be hesitant to participate. This is often an issue with juniors.
- Finally, a company culture that does not emphasize the importance of collaboration and continuous improvement may discourage Developers from participating in refinement sessions, as in "Developers code, they do not write tickets." This attitude is a common issue in the early stages of transformations to Scrum in more traditionally minded organizations.

Remedy: The idea that Developers need to spend significant time on Product Backlog refinement is not an easy sell. Often, the organization and Developers themselves have a limited understanding of what the engineers contribute to value creation:

- Many Developers want to solve puzzles; they do not care that much about the origin of the puzzles.
- Managers may not want Developers to do work unrelated to creating Increments.

Whatever the reason for opposition, I suggest running an experiment in which the Developers—at least those who are interested in participating—support the Product Backlog refinement process. Then compare the results in a Retrospective, probably including line management and stakeholders, to convince everyone of the usefulness of Developers refining Product Backlog items.

30. Too Much Refinement

Observation: The Scrum Team has too many refinement sessions, resulting in a too-detailed Product Backlog.

See Chapter 10, anti-pattern 21, for a detailed description.

31. Too Much Estimating

Observation: During the creation of the Sprint plan, the Developers estimate Product Backlog items from the Sprint Backlog down to the level of sub-tasks.

See Chapter 11, anti-pattern 6, for a detailed description.

32. Submissive Engineers

Observation: The Developers submissively follow all demands of the Product Owner.

Background: Challenging the Product Owner on whether the contents and order of the Product Backlog will result in the greatest value creation is the noblest obligation of every team member: Why should we do this?

Success with Scrum relies upon everyone fulfilling the checks and balances built into the framework. It is easy to fall in love with your solution, neglecting to address the real customer problem. If Developers simply build whatever the Product Owner suggests, the value created by the Scrum Team will likely fall short of its potential. That is why you want team members to have a healthy skepticism, willing to question everything to identify the best possible solution under the given circumstances.

Remedy: There are several aspects to this issue:

- **Fear of conflict:** The Developers might avoid challenging the Product Owner due to fear of conflict or confrontation, preferring to maintain a harmonious working environment. This motivation points to issues with living Scrum Values, particularly regarding psychological safety.
- **Lack of confidence and overreliance on the Product Owner:** Some Developers may need more confidence to question the Product Owner's decisions, doubting their understanding of the customer's needs or the product's requirements. Here, it would help to have Developers talk directly to customers, for example, during user interviews.

- **Hierarchy and authority:** Developers might perceive the Product Owner as a management position and feel obliged to follow their instructions without questioning them. The point is that there are no managers in Scrum; no one has any authority to tell anyone else what to do.
- **Apathy or disengagement:** In some cases, Developers might be disengaged or lack motivation, leading them to follow the Product Owner's instructions without questioning them. The effect is a classic outcome of disempowered teams directed by outside stakeholders. Stripping self-management off teams turns skeptics into cynics—if they keep their jobs in the first place.
- **Time pressure:** Tight deadlines or external pressures might lead Developers to prioritize getting work done over challenging the Product Owner's decisions. That is a typical side effect if organizations impose (arbitrary) deadlines on teams.
- **Focus on technical aspects:** Some Developers might be more interested in the technical aspects of the product than in its business value or customer impact, making them less likely to question the Product Owner's choices.

This anti-pattern requires time, dedication, and perseverance to solve, as it points to multiple dysfunctions in the organization. You can start the necessary change process with the following first steps:

- **Promote Scrum Values:** Scrum Masters should act as role models for Scrum values. These values foster a culture of questioning, constructively handling disagreements and conflicts, and continuous improvement.
- **Empower Developers:** By training and coaching the Developers to understand the customer's needs and product requirements, Scrum Masters can boost their confidence to challenge decisions. For example, a first step could involve facilitating customer interactions or user interviews.
- **Cultivate equality:** Scrum Masters need to reiterate that there are no hierarchies in Scrum and that everyone's input is valued. They can reinforce that the Product Owner's role is to maximize value, not to command and control "their" Developers.
- **Promote engagement:** To deal with apathy, the Scrum Master can help the Developers understand their crucial role in product development. Putting them in touch with real users to learn about how their work solved the users' problems is a helpful step.

- **Mitigate time pressure:** To lessen the impact of tight deadlines, Scrum Masters can help with realistic Sprint planning and emphasize that understanding the why behind decisions can save time in the long run.
- **Shift focus to value:** Scrum Masters can guide Developers to consider the business value and customer impact in addition to technical aspects during Product Backlog refinement and Sprint planning.

Food for Thought

Is Scrum too egalitarian? There are no hierarchies, subroles, or silos within a Scrum Team but a collective responsibility on the Developers' side. How can we then align the apparent need for distinguishing accomplishment levels with Scrum's egalitarian nature? The same applies to topics like software architecture and team topology.

Conclusion

Given their essential role in making Scrum successful, it is no surprise that there are plenty of Developer anti-patterns to observe. The good news is that many of them are entirely under the control of the Scrum Team. All it takes to tackle these anti-patterns for the team is starting to inspect and adapt, addressing one improvement area at a time. Why not use your next Retrospective for this purpose?

Toolbox

Regarding the following tips and exercises, you will find a detailed description in Appendix B:

- **"20 Questions from a New Scrum Master to the Developers"** comprises 20 questions addressing the foundations of a Scrum Team's capability to build valuable products: from technical excellence and what it takes to achieve this high proficiency level to technical debt.
- **"Agile Metrics—The Good, the Bad, and the Ugly"** points to suitable agile metrics reflecting either a Scrum Team's progress in becoming agile or an organization's progress in becoming a learning organization.

- **"Building Communities of Practice"** helps win hearts and minds within the organization. It provides authenticity to the agile transformation—signaling that the effort is not merely another management fad.
- **"Creating Product Goals as a Collaborative Practice"** suggests using Ralph Jocham's Product Goal Canvas to create a shared understanding among the Scrum Team members about the why, what, and how of the team's journey.
- **"Daily Scrum with Distributed Teams"** points to a set of Liberating Structures that support distributed or hybrid teams in organizing a meaningful Daily Scrum.
- **"How to Create a Definition of Done"** details a collaborative exercise of the whole Scrum Team, probably including some of the stakeholders at a later stage, to create, inspect, and adapt a Definition of Done.

4 Scrum Stakeholder Anti-Patterns

Introduction

While stakeholders are not part of the Scrum Team and are not even an official role in Scrum, they have a tremendous influence on how the Scrum Team works and on whether or how they make progress toward their Product Goal. Effective stakeholders help the Scrum Team deliver value to customers, while ineffective stakeholders act as if the purpose of the Scrum Team is to serve their stakeholders; see Figure 4.1.

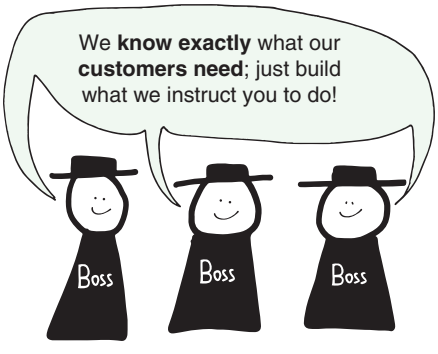

Figure 4.1 In a complex environment, there are no experts. Consequently, pretending to know exactly how to solve your customers' problems defies decades of practical experience in many organizations.

COMMON SCRUM STAKEHOLDER ANTI-PATTERNS

Scrum stakeholder anti-patterns are typically driven by one or more common root causes:

- **Fear of change:** Stakeholders in the organization may fear losing power, control, or comfort with the familiar. The Agile Manifesto values "individuals and interactions over processes and tools,"[1] but this shift could mean abandoning existing hierarchies, which may make stakeholders anxious.
- **Siloed organizational structures:** Functional silos are critical elements of the legacy organizational design, reflected in many stakeholders' status and remuneration. Preserving them for political reasons hinders cross-functional collaboration, an essential aspect of Agile.
- **Misaligned incentives:** The organization may still have management incentives tied to traditional performance measures that are incompatible with Scrum and agile principles.
- **Short-term focus:** Legacy organizations often focus on immediate results, which clashes with Scrum's focus on long-term sustainability.
- **Resistance to learning:** Becoming agile requires unlearning old practices and learning new ones. The effort required can be intimidating. Additionally, there may be an insufficient training and coaching offering to stakeholders.

As a result, the organization may limit an agile transformation to product and engineering teams while the rest continue practicing the industrial-age principles.

The following list of Scrum stakeholder anti-patterns addresses system-related issues, issues of individual players, and anti-patterns specific to Scrum events.

SCRUM STAKEHOLDER ANTI-PATTERNS AT THE ORGANIZATIONAL LEVEL

1. Lack of Transparency

Observation: The organization is not transparent about vision and strategy; hence, the Scrum Teams are hindered from becoming self-managing.

1. Kent Beck, Mike Beedle, Arie Bennekum, et al., "The Agile Manifesto," Agile Alliance, 2001, https://agilemanifesto.org.

Background: "If you don't know where you are going, any road will get you there."[2] In a complex environment with high levels of competition and uncertainty, everyone needs to understand the why, the what, the how, and the who. A lack of transparency will stop any effort to become agile dead in its tracks, as those interested in change likely won't flock to the cause. Elon Musk's "The Secret Tesla Motors Master Plan"[3] provides an excellent example of leadership transparency at the vision and strategic levels.

There are some reasons for this secrecy:

- **Legacy hierarchical structure:** In traditional hierarchical structures, information flow is often top-down. The notion of transparency across all levels can be unfamiliar and uncomfortable.
- **Fear of losing control:** Leadership may feel that by revealing their vision and strategy, they will lose control over the company's direction.
- **Fear of information misuse:** Leadership might fear that transparency about vision and strategy could lead to misuse of information by competitors or even employees, perpetuating the information principle of sharing on a need-to-know basis.[4]
- **Perceived inefficiency:** Leaders may believe that sharing strategic information with everyone could lead to debates, slowing down decision-making processes.
- **Uncertainty or lack of clarity:** Sometimes, the leadership may not have a clear vision or strategy. Hence, they cannot communicate what they don't fully understand, leading to a lack of transparency.

Remedy: While the Scrum Guide does not provide any information on transforming legacy organizations into agile organizations, practices that can help to overcome the lack of transparency at the leadership level include the following:

- **Promoting transparency:** Actively advocate for a more transparent environment, and use the Scrum values, particularly openness, as a basis for this argument.

2. Lewis Carroll Quotes, BrainyQuote.com, 2023, https://www.brainyquote.com/quotes/lewis_carroll_165865.
3. Elon Musk, "The Secret Tesla Motors Master Plan (Just between You and Me)," Tesla, August 2, 2006, https://www.tesla.com/blog/secret-tesla-motors-master-plan-just-between-you-and-me.
4. The *need-to-know basis* is a principle in information security whereby sensitive information is disclosed only to those individuals who require the data to perform their duties effectively. This approach limits the risk of unauthorized or unnecessary disclosure, helping to protect intellectual property, prevent data breaches, and maintain operational security.

Encourage leadership to understand the importance of sharing the company's vision and strategy.

- **Facilitating communication:** Facilitate regular meetings where executives can share strategic updates beyond Scrum events. Also, encourage Developers and the Product Owner to address questions directly to the leadership team to clarify the direction and align on the objectives.
- **Influencing through education:** Provide training or workshops to the leadership team. Highlight how transparency can empower teams, increase engagement, and ultimately drive better product outcomes.
- **Visualizing strategy:** Use artifacts like a product roadmap[5] to visualize the strategy and connect it to the team's Product Goal, its Product Backlog, and its Sprint Backlog, making the alignment of the daily work with the big picture more tangible.
- **Celebrating success:** Publicly recognize when transparent behavior leads to positive outcomes, encouraging more people in the organization to see and adopt the benefits of transparency.

2. Lack of Leadership Support

Observation: Senior management is not participating in agile processes, for example, the Sprint Reviews, despite being a role model. Instead, they expect a continuation of the previous reporting.

Background: The support of the upper management is mission-critical for any meaningful application of Scrum; see Figure 4.2. No Scrum Team will be successful if the leadership is not leading the effort—for example, by attending Sprint Reviews to signal to everyone in the organization to take the effort seriously.

5. A product roadmap is a strategic document outlining the vision and direction of a product over time. It details key features, expected milestones, and targets, serving as a communication tool between stakeholders and the product team. To understand its effective creation and usage, consider reading *Product Roadmaps Relaunched: How to Set Direction While Embracing Uncertainty* by C. Todd Lombardo, Bruce McCarthy, Evan Ryan, and Michael Connors (O'Reilly Media, 2017).

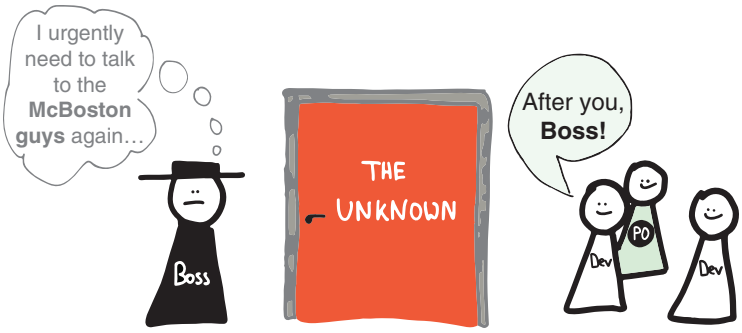

Figure 4.2 Transformations require the unwavering support of the leadership team, particularly when the path to success is uncertain.

Remedy: There are ways we can convince the leadership team to "lead" the change effort:

- **Create a shared vision:** Help leaders create a shared vision for applying Scrum, and articulate why it matters for the company. This way, they have a story to tell their teams, helping to drive engagement and buy-in.
- **Ask for their support:** Don't be shy about directly asking for their involvement. Explain why their participation is critical and how it can benefit the entire organization.
- **Show Agile's value:** Use data to showcase how adopting agile practices can deliver better results. Use actual case studies, either from within the company or from successful agile transformations in similar companies.
- **Peer influence:** Arrange for leadership to meet with executives from other companies who have successfully undergone agile transformations. These peers can share their experiences and insights, which might be more persuasive.
- **Involve leaders:** Invite them to actively participate in events like Sprint Reviews, giving them first-hand experience of Scrum in action and the benefits it can bring.
- **Change the reporting style:** Use Scrum artifacts like a Product Backlog or Sprint Backlog to demonstrate progress instead of traditional reports, making the agile processes more tangible for them.
- **Train and educate:** Provide Agile and Scrum training sessions specifically tailored for leaders, helping them understand the value, principles, and practices of agile and how they can lead by example.

3. Cherry-Picking Individual Practices

Observation: Practices are either bent or ignored whenever it seems appropriate; for example, the Product Owner role is reduced to a project manager role. Or stakeholders are bypassing the Product Owner to get things done and get away with it in the eyes of the senior management, as they would show initiative. There is a lack of discipline to support the agile transformation.

Background: Cherry-picking practices while ignoring the context required to make them work is a common form of window dressing agility. However, being a part of a complex system, isolated elements seldom work if the organization aims to achieve business agility.

Some reasons why an organization may believe that picking isolated agile practices will lead to the desired change at a system level are as follows:

- **Misunderstanding agile practices:** Organizations may misunderstand the agile mindset as a set of tools or practices rather than a comprehensive approach based on an overarching philosophy requiring cultural change. This misconception may lead to the selective implementation of techniques without the necessary mindset shift.
- **Resistance to change:** Organizations may resist sweeping change for fear of disruption or uncertainty. Implementing isolated agile practices may seem like a safer, incremental approach to change.
- **Short-term focus:** Organizations may focus on immediate results or quick wins, believing that implementing certain agile practices will yield immediate benefits. They might overlook the longer-term systemic changes necessary to embrace Agile and Scrum truly; see Figure 4.3.
- **Lack of training and knowledge:** Without sufficient knowledge of agile principles and how they interconnect, organizations may not fully understand the importance of implementing all aspects of Scrum.
- **Hierarchical culture:** Traditional hierarchical cultures may resist the distributed decision-making inherent in Agile, causing them to cherry-pick practices that fit more comfortably within their existing structure.

Remedy: Embracing agile practices such as Scrum becomes considerably simpler when organizations are driven by a concrete objective, such as “we must improve

our responsiveness to customer needs consistently, or we risk going out of business." Simply having retrospectives once in a while won't cut it.

Figure 4.3 The command and control-induced obsession with forecast and commitment matching delivery.

So, what practical, hands-on steps can a Scrum Master as a change agent use to try to change their organization's attitude of picking isolated practices instead of accepting large-scale change?

Some first steps may include the following:

- **Educate:** Organize workshops, training sessions, and interactive games, for example, the Agile Fluency® game,[6] to illustrate the benefits of adopting a holistic, agile approach. Use examples and case studies of successful agile transformations to make a compelling argument.
- **Visualize the benefits:** Map out how a complete Agile transformation can benefit the organization in the long term. For instance, you could use the concept of a value stream to show how full Agile adoption can improve workflow and add value to the product or service.

6. According to the Berlin Product People, "The Agile Fluency Game™ is a simulation of how a team adopts agile practices, referring to the Agile Fluency® Model, developed by James Shore and Diana Larsen." For more information, see https://berlin-product-people.com/services/agile-fluency-game.

- **Incremental change:** Start small, with one team or project, and show the benefits of a complete agile transformation. Use this as a case study to push for broader change. Be sure to celebrate small wins along the way to build momentum.
- **Provide a safe environment for experimentation:** Encourage teams to try new agile practices and allow for failure. Use team and stakeholder Retrospectives to learn and adapt, promoting a culture of continuous learning and improvement.
- **Foster collaboration:** Encourage collaboration between different roles and departments, supporting the breakdown of silos and demonstrating the benefits of working as a cross-functional team.
- **Live the agile values:** As a Scrum Master, exemplify Scrum values and principles in your daily work. Be transparent, inspect and adapt, and continuously strive for improvement. Your actions can motivate others to do the same.

4. Scrum on a Tight Budget

Observation: The organization does not spend adequate time and budget on proper communication, training, and coaching to create a shared understanding of purpose and direction among all organization members about Scrum.

Background: Scrum can't simply be decreed and rolled out, even if the process is outsourced to a third party, such as a consulting company. Consequently, becoming agile and embracing business agility is a monumental and expensive undertaking. It requires fostering a culture that values collaboration, responsiveness to change, and continuous improvement to reap a competitive advantage.

Moreover, while these expenditures become immediately apparent, the organization may not reap the return on investment for years to come, and cutting back on necessary expenses early in the transformation will make failure more likely.

Why might an organization embark on becoming agile and embracing business agility but also sabotage itself by not allocating sufficient financial means to accomplish its intention? Some of those reasons may be as follows:

- **Misunderstanding of Agile:** Organizations might perceive Agile and Scrum as merely process frameworks that don't require an extensive financial commitment. They underestimate the financial investment needed to overhaul practices, culture, and mindset.

- **Short-term focus:** Leadership may prioritize immediate financial results over long-term strategic benefits, confusing "becoming agile" as a path to score a short-term gain.
- **Budget constraints:** Economic challenges or existing financial commitments may lead to underinvestment in an agile transformation. The organization may believe it can make do with less.
- **Perceived as a cost center:** Some organizations view expenses related to agile transformation (training, coaching, tooling) as a cost center, not an investment in future productivity and adaptability.
- **Resistance to change:** Some leaders may hesitate to allocate significant resources to a transformation they don't fully understand or believe in.
- **Overconfidence:** The organization might believe its employees can manage the transition without extensive support, training, or coaching, overlooking the need for expert guidance and the time it takes for people to adapt to new ways of working.
- **Undervaluing communication and training:** Organizations may underestimate the importance of effective communication and education in a successful transformation, dismissing those as soft skills.

Remedy: Here are some approaches to deal with insufficient funding for your transition to Scrum:

- **Pitch the value of Agile:** Help the leadership understand the tangible benefits of agile transformation, such as faster time to market, higher customer satisfaction, increased team productivity, and adaptability to change.
- **Get an ally:** Find a sponsor or a champion in the leadership who understands the value of an agile transformation and is willing to advocate for it.
- **Present case studies:** Share successful agile transformation stories from similar companies or industries. Highlight the investments these organizations made and the benefits they reaped over time.
- **Facilitate learning:** Host workshops or seminars on agile and Scrum for the leadership. Invite experienced consultants or experienced agile practitioners who can provide insights into the journey, the investment needed, and the potential return on investment.

- **Propose a pilot program:** If the organization hesitates to invest heavily, suggest a pilot agile transformation with one Scrum Team, allowing the organization to see first-hand benefits without a massive upfront investment.
- **Show incremental improvement:** Adopt an iterative approach to demonstrating improvements. Start with small, inexpensive changes showing potential, then build on those successes to justify the more significant investment. For example, Sprint Review events of your pilot are well-suited for that purpose.

As per the Scrum Guide, the role of a Scrum Master includes promoting and supporting Scrum by helping everyone understand Scrum theory, practices, rules, and values. This role concept includes that the organization understands the investment needed to reap the full benefits of a transformation to Scrum.

5. Telling People How to Do Things

Observation: The stakeholders, namely the managers or team leads, also detail how the Scrum Team shall solve an issue and create value.

Background: In the good old days on an assembly line, it was a valuable skill for managers to be able train new workers on the job, just as the managers probably were trained when they worked on the assembly line. Nowadays, as we invest most of our time building products that have never been built before, this attitude becomes a liability.

However, if your stakeholders persist in this behavior, consider investigating whether it stems from one of these potential reasons:

- **Perceived risk:** If the importance of the issue at hand for the career of the respective stakeholder is high, they may believe in the necessity to control the solution to mitigate risks. They may perceive their involvement as ensuring the Scrum Team doesn't make mistakes.
- **Ego or status:** Some stakeholders might feel that providing solutions reinforces their organizational status or expertise. They may view their role as problem solvers and might feel threatened by the idea of a self-organizing team.
- **Legacy thinking:** In many traditional organizations, managers are used to providing solutions instead of just defining the problem, pointing to the legacy of the hierarchical, top-down management style where decision-making is centralized.

- **Misunderstanding of Agile and Scrum:** Stakeholders may not fully understand the essence of agile and Scrum, where we entrust self-organizing teams with finding the best solution.
- **Lack of trust:** Stakeholders may lack trust in the ability of the Scrum Team to solve problems effectively without their input. This trust issue could be due to past experiences, a new or inexperienced team, or simply an unfounded fear.
- **Misinterpretation of communication:** Stakeholders may consider their message to the Scrum Team as providing input or ideas, while the team perceives it as directives.

Remedy: What practical steps, beyond educating your managers, can you take to convince them that stating the problem and providing the required means is sufficient for a self-managing Scrum Team to identify a solution to the problem? Some suggestions are as follows:

- **Facilitate collaboration:** Generally, foster a culture where stakeholders and the Scrum Team can regularly exchange ideas and insights but where the final decisions on how to achieve the Sprint Goals rest with the team.
- **Promote trust building:** Facilitate activities such as Sprint Reviews that help build trust between stakeholders and the Scrum Team. Share past success stories where the team successfully delivered solutions, making the process visible and transparent.
- **Coach one-to-one:** Engage resistant stakeholders individually to understand their concerns and coach them on the benefits and principles of agile and Scrum. This personalized attention can often address specific misunderstandings or fears.
- **Practice Scrum:** Make sure that stakeholder requests go through the Product Owner, reinforcing the Scrum rules to help stakeholders understand their role. In a complex environment, there are no experts: "How this is done is at the sole discretion of the Developers. No one else tells them how to turn Product Backlog items into Increments of value."[7]
- **Change manager incentives:** Reward managers' results in helping their Scrum Teams to self-manage instead of rewarding how well they manage or regulate those teams.

7. Ken Schwaber and Jeff Sutherland, "Sprint Planning," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#sprint-planning.

6. Steering Committee Meetings

Observation: Unimpressed by the agile ways of working, the stakeholders insist on continuing the biweekly steering meetings to ensure that the Scrum Team will deliver all their requirements on time.

Background: Scrum does not comprise a steering committee because it focuses on self-managing teams and direct accountability. The Product Owner is the individual in charge of product-related decisions, using their profound understanding of problem and solution space to formulate and communicate a Product Goal, which is reflected in the composition and the ordering of the Product Backlog. Consequently, the absence of a steering committee streamlines decision-making and negates the delay caused by seeking agreement from numerous parties, which is beneficial in a complex environment.

Stakeholders may insist on steering meetings due to fear of change, lack of understanding of agile principles, distrust in the Scrum Team, misaligned objectives, ambiguity over accountability, and concerns about transparency. These reasons typically stem from comfort in familiar processes and apprehension toward self-management and direct responsibility in Scrum. In this respect, the motivation for this anti-pattern overlaps with other reasons stated earlier.

Remedy: You can accelerate the pending change in your organization by multiple practices; for example:

- Educating stakeholders on Scrum's principles
- Sharing success stories from other organizations
- Fostering trust between stakeholders and the Scrum Team by regularly delivering valuable Increments
- Increasing transparency through Scrum events
- Challenging previous "valuable requirements" with data regarding their real value

If necessary, you can propose a gradual reduction of steering committee meetings while increasing the Scrum Team's autonomy, demonstrating the benefits of this shift over time. This approach aims to empower the team while building trust, not to eliminate oversight. Call it an incremental approach to introducing Scrum.

7. Talking to Customers Is Off Limits

Observation: The sales organization and other functional silos guard the direct access to customers, thus preventing the Scrum Teams from learning directly from customers.

Background: The sales organization probably deems direct contact of Scrum Team members with customers too risky (for their sales plans) and prevents it from happening.

The problem is that filtering feedback from customers mainly results in a limited understanding of the customer's situation on the side of the Scrum Team. Furthermore, due to a likely lack of technical competence on the side of the sales organization, critical information regarding potential solutions may never reach the Scrum Team, thus limiting its ability to tailor a solution to the customer's needs.

You may ask yourself why stakeholders such as the salespeople prevent Scrum Team members from directly communicating with customers. There are some possible reasons beyond the usual misunderstanding or principles or lack of trust; for example:

- **Fear of losing control:** Sales teams, used to being gatekeepers of client communication, may fear losing control or relevance if the Scrum Team interacts directly with the customers. They might feel threatened that their roles could be marginalized if the customers start getting comfortable interacting with the Scrum Team.
- **Misalignment of goals:** Salespeople typically focus on closing deals and hitting quotas. They may worry that if Developers interact directly with the customers, it might lead to promises or discussions that don't align with the sales strategies or targets.
- **Concern over professionalism and communication skills:** Sales teams may believe Developers lack the communication skills or industry knowledge to interact effectively with customers. They may worry about technical jargon alienating the customer or that Developers might inadvertently share sensitive project information.
- **Risk aversion:** Sales teams may also worry that direct contact could lead to the exposure of internal processes or conflicts that could damage customer relationships or increase customer expectations.

Remedy: To become an effective product creation organization, it is essential that Scrum Team members directly communicate with customers regularly. But how can you persuade stakeholders that fostering a direct dialogue between Scrum Team members and customers is a win-win situation where Developers gain immediate insight into customer challenges?

Here are a few strategies to accomplish this goal beyond generally educating the stakeholder on agile practices and principles:

- **Facilitate joint meetings:** Initially, you can propose joint meetings where salespeople and the Scrum Team interact with customers. This approach can help alleviate fears and build trust over time while ensuring the Scrum Team gets the necessary insights.
- **Develop communication skills:** Encourage Scrum Team members to improve their communication skills, for example, by employing a communication trainer. A team that can communicate well can reassure stakeholders that they can handle direct customer interactions appropriately.
- **Respect boundaries:** Ensure that the Scrum Team understands and respects the boundaries of customer interaction, which can help alleviate the fears of stakeholders. For instance, the Scrum Team should realize that they shouldn't promise customers product features or delivery timelines but instead should refer them to the Product Owner.
- **Regular updates:** Propose a feedback loop where the Scrum Team regularly reports to the stakeholders about customer interactions, what team members learned, and how it impacts the work. This transparency can build trust and assure stakeholders that the process works as intended.

SPRINT ANTI-PATTERNS OF THE IT MANAGEMENT

Those stakeholders closest to the Scrum Teams—the IT management—also display some anti-patterns.

8. All Hands to the Pumps without Scrum

Observation: The management temporarily abandons Scrum in a critical situation; see Figure 4.4.

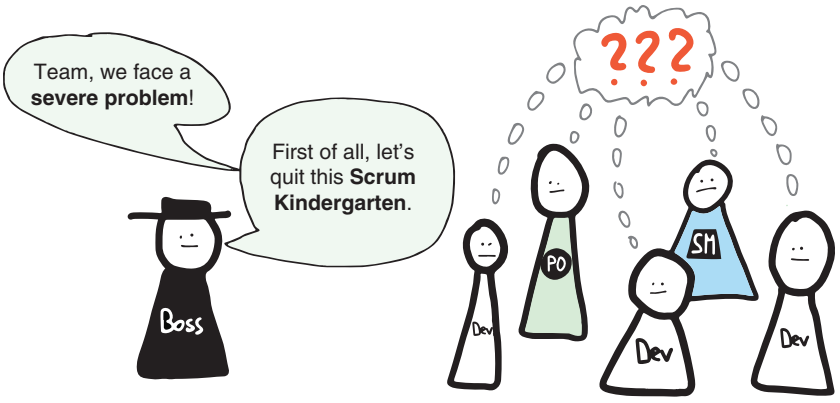

Figure 4.4 All too common: Abandoning Scrum in a moment of crisis instead of trusting the Scrum Team to identify a solution.

Background: This is a classic manifestation of disbelief in agile practices, fed by command-and-control thinking. Instead, canceling the Sprint and gathering the Scrum Teams would likely solve the issue.

However, managers may be tempted to pull rank in such a situation for several reasons; for example:

- **Bias for action:** In a lack of patience, managers may tend toward immediate action in crises, feeling that Scrum's empirical, self-managed process slows response time.
- **Fear of personal failure:** Managers may believe failure is more likely without exercising control, considering traditional command-and-control methods and clear communication structures are more reliable.
- **Misunderstanding of Scrum:** Misconceptions about Scrum can lead managers to abandon it under pressure, considering it incompatible with crisis management.
- **Influence of power structures:** Hierarchical power structures might prompt managers to seize control in crisis situations, neglecting Scrum's self-management principle.

Remedy: There is no shortcut to overcoming this anti-pattern. Pointing to the Scrum Guide in a situation where your line manager probably fears for their career will accomplish nothing. Instead, cultural change requires time and perseverance. Here are some approaches that may help:

- **Highlight Scrum's empiricism:** Show that Scrum's flexibility, inspection, and adaptation mechanisms are particularly effective in managing crises.
- **Foster trust:** Help managers build trust in Scrum Teams' ability to self-manage, including during crises. Showcase examples where Scrum Teams effectively handled crises.
- **Engage management:** Involve managers in Scrum events such as Reviews and Retrospectives to help them better understand Scrum's effectiveness and the team's capabilities.
- **Emphasize learning from failure:** Reinforce that failure is a part of the learning process in Scrum and a valuable source of insights for improvement.

9. Reassigning Team Members

Observation: The management moves team members from one Scrum Team to another.

See Chapter 5, anti-pattern 23, for a detailed description.

10. Team Autonomy Undermined

Observation: A manager assigns specific tasks directly to Developers, thus bypassing the Product Owner and ignoring the Developers' self-management prerogative. Alternatively, the manager removes a Developer from a team to work on such a task.

See Chapter 5, anti-pattern 24, for a detailed description.

INCENTIVIZED SCRUM STAKEHOLDER ANTI-PATTERNS

There are numerous ways in which stakeholders can impede the progress of a product team. Five of the most common ones are listed in this section.

11. "My Budget" Syndrome

Observation: Stakeholders do not pitch for a Scrum Team's attention but claim that they allocate "their" budget on feature requests as they see fit, ignoring the Product Owner's prerogative to define Product Goals and manage Product Backlogs.

Background: As mentioned before, the Product Owner is the one who handles all decisions about the *product*. They use their deep understanding of the issues and

potential solutions to create and share a Product Goal, which is visible in the makeup and order of the Product Backlog. Effectively, Scrum separates budgetary authority from making product decisions.

Conversely, the "My Budget" Syndrome anti-pattern attempts to preserve that product decision-making authority on the side of the stakeholders, defying Scrum's purpose and leading to the pursuit of local optima at a silo or departmental level. Here, stakeholders may be guided by a fear of losing control, a lack of understanding of agile practices, siloed thinking, traditional performance metrics, and a general lack of trust in Scrum Teams. Often, this effect can be observed in organizations that tie additional benefits or incentives to the individual stakeholder. (Note: Pet projects also fall into this category.)

Remedy: Generally, product development capacity needs to be allocated in the spirit of optimizing the return on investment for the whole organization. Scrum supports this approach by separating the budgetary from the product decision-making authority.

To counter this anti-pattern, you could consider initiating the following actions:

- **Change advocates:** Identify and work with champions within the stakeholders who understand and support the Scrum approach. They can help influence their peers to accept the change.
- **Incremental change:** Encourage stakeholders to gradually relinquish control over a few small-scale projects to observe how Scrum functions and build trust in the process.
- **Education and training:** Facilitate workshops to explain the role of the Product Owner and the concept of self-managing teams to stakeholders. Show how Scrum decentralizes decision-making, leading to more customer-centric, profitable products.
- **Promote collaboration:** Encourage regular interactions between stakeholders and the Scrum Team. Invite stakeholders to Sprint Reviews to give them a voice, include them in the decision-making process, and generally help them see the value created by the team.
- **Realign metrics and incentives:** Work with the people management or other relevant departments to re-evaluate and redesign performance metrics and incentives that align with Scrum principles, encouraging cooperation.

12. "We Know What to Build"

Observation: Stakeholders see no benefit in any interaction of the product delivery organization with customers, as they have precise ideas of what will be successful with customers.

Background: Several reasons may cause this phenomenon; for example:

- A founder or entrepreneur pursues their product vision without engaging in customer discovery activities.
- The product delivery organization is solely briefed indirectly by key account managers.
- The sales department probably deems direct contact of the Scrum Team members with customers too risky and prevents it from happening.

What are the reasons that some stakeholders believe they can predict the product's future to fame and glory when no one can predict the future in a complex environment? Some motivations might be, for example:

- **Overconfidence bias:** Stakeholders, particularly those in leadership positions, may fall into the overconfidence trap, assuming their experience, knowledge, and intuition can accurately forecast the future.
- **Illusion of control:** Stakeholders may feel an exaggerated sense of control over the product's outcome due to their involvement in the project or authority in the organization.
- **Hindsight bias:** Past successes often lead to false confidence in predicting future outcomes.
- **Confirmation bias:** Stakeholders may selectively interpret information to validate their preconceived notions about the product's success, ignoring or discounting information that contradicts these beliefs, starting with preventing it from being gathered.
- **Self-serving bias:** This bias leads stakeholders to attribute successful outcomes to their capabilities and decisions while blaming external factors for failures. This skewed perception can feed their illusion of being able to predict the future accurately.
- **Resistance to change:** Stakeholders may also resist agile processes as they shift from a predictive (plan-based) approach to an adaptive (iterative) one. They

might cling to predictive mindsets due to comfort with standard methodologies or fear of uncertainty.

Remedy: So, what can we attempt to convince stakeholders that they cannot predict the product's future more accurately than anyone else in a complex environment? Moreover, that instead, Scrum's iterative and incremental approach with autonomous, empowered teams is best suited to solve customers' problems and maximize the return on investment?

Some steps, for example, are as follows:

- **Leverage customer feedback:** Arrange for sessions where stakeholders can hear directly from customers. Hearing how iterative delivery brings value and fosters improvement based on customer feedback can help convince them of the unpredictability of the product's future.
- **Counsel on the dangers of overconfidence:** Gently remind stakeholders of the cognitive biases that can impair judgment and decision-making. It's crucial to foster a culture where it's okay to be wrong and to learn from errors rather than rigidly sticking to an initial hypothesis about the product's future.
- **Facilitate stakeholder workshops:** Organize interactive workshops or seminars where stakeholders can learn about Scrum, its benefits, and its impact on return on investment. Include hands-on exercises that simulate Scrum work processes to provide them with a first-hand iterative development experience.
- **Encourage stakeholder involvement:** Invite stakeholders to Sprint Review sessions, helping them understand the work, see the progress, and realize the complexity and uncertainties involved, demonstrating how Scrum Teams adapt to changes and manage uncertainty better than traditional approaches.
- **Showcase metrics and progress:** Regularly share key performance indicators, metrics, and progress with stakeholders. Evidence of steady progress and value delivery can convince them about the efficacy of the iterative approach in handling complex projects.

13. Selling Nonexistent Features

Observation: What features do you need us to provide to close the deal? Sales managers chase sales objectives by asking prospective customers for a feature wish list and forwarding those to the Scrum Team as requirements.

Background: Pursuing the sales process in such a way often leads the product into a feature comparison race to the bottom, inspired by bonuses and personal agendas.

The problem with asking prospective customers is that they usually lack the depth of knowledge required to provide valuable answers to this question. Moreover, they also often lack the level of abstract thinking necessary to develop a valuable, viable, usable, and feasible solution. As the saying goes, if the only tool you are familiar with is a hammer, every problem will look like a nail. This challenge is why product folks like to observe customers in their typical environment to identify their problems and potential solutions.

Remedy: What can you do to convince salespeople to stop violating basic agile principles by selling nonexistent features without anyone from the Scrum Team, particularly the Product Owner, knowing?

Following are some first steps:

- **Foster collaboration between the sales team and Scrum Team:** The sales team needs to understand the importance of selling the value the product brings rather than selling specific nonexistent features. Regular communication and collaboration between the two teams promotes that understanding.
- **Involve salespeople in Scrum events:** Invite salespeople to Sprint Reviews to help them understand the current Increment and what is in the Product Backlog for future Sprints. This visibility prevents them from overpromising features that are unavailable or won't be available in the near future.
- **Develop a joint sales–Scrum charter:** Develop a charter or agreement outlining how sales and the Scrum Team will work together, including principles such as "We won't sell what we don't have" and "We will consult with the Scrum Team before making promises to clients about features."
- **Create a product roadmap:** A high-level product roadmap can provide the sales team with a view of planned features. Ensure the roadmap is flexible and clearly states that changes based on customer feedback and market conditions may occur.
- **Show the impact of selling nonexistent features:** Use examples or case studies to show how selling nonexistent features can lead to customer dissatisfaction, loss of credibility, and decreased sales in the long run.

- **Provide a feedback channel:** Set up a process for salespeople to communicate customer wishes and feedback to the Scrum Team and the Product Owner. This channel could be a regular meeting or an online platform where they can share their insights.
- **Promote a solution-oriented mindset:** Encourage salespeople to focus on solving customer problems rather than selling features. This mindset involves understanding customer needs and aligning the product's value proposition with these needs.

14. Financial Incentives to Innovate

Observation: The organization incentivizes generating new product ideas and suggestions monetarily.

Background: Contributing to the long list of ideas and thus to the short list of hypotheses should generally be intrinsically motived. A moderate form of recognition for contributions seems acceptable, particularly at a social level. However, it should not be so substantial that it might trigger unintended consequences; beware of a signal-to-noise ratio dilemma.[8] For more information, please also watch the webinar Product Discovery Anti-patterns.[9]

Remedy: What can we do to avoid drowning in low-value product suggestions? Consider starting with the following:

- **Clear guidelines:** Provide clear and precise guidelines for the kind of suggestions you're looking for. Ensure participants understand the organization's objectives, context, and specific problems to solve.
- **Training and education:** Equip teams with tools, resources, and training to generate high-quality suggestions, for example, regarding Design Thinking and other product discovery practices.

8. The signal-to-noise ratio is the relationship between valuable suggestions (signal) and irrelevant or non-beneficial suggestions (noise). When incentives encourage many contributions, they may result in an overwhelming quantity of low-quality ideas that do not significantly contribute to innovation. This effect dilutes the valuable suggestions, making it difficult for Product Owners to identify and act on them, thus not resolving the original problem and potentially creating new issues.
9. Stefan Wolpers, "Product Discovery Anti-Patterns (Hands on Agile Webinar #1)," Age-of-Product.com, 2017, YouTube video, 22:51, https://youtu.be/rW2kVanMxMM.

- **Quality over quantity:** Instead of rewarding the sheer number of suggestions, establish an incentive system that values the quality of the ideas, their feasibility, and their potential impact.
- **Peer review:** Implement a peer-review process where two random organization members evaluate each other's suggestions. This process encourages thoughtful contributions and reduces the number of low-quality suggestions.
- **Iterative feedback:** Provide continuous and constructive feedback to the participants to help them improve the quality of their suggestions over time.
- **Dedicated review team:** Have a dedicated team to review and filter the suggestions. This team should be trained to recognize good ideas and deeply understand the organization's goals and values.
- **Proper channels:** Have transparent channels for submitting ideas, including a dedicated platform or software where contributors can suggest ideas, discuss them, and follow their status.

STAKEHOLDER ANTI-PATTERNS AT SCRUM EVENT LEVEL

The Sprint

Anti-patterns of this sort point at stakeholders' ignorance of the core idea of Scrum—self-managing teams.

15. (Regular) Emergency Work

Observation: Someone in your organization has sold a nonexistent feature or functionality to a customer to close a deal, probably already including delivery dates and contractual penalties for nondelivery. Now, they want the Scrum Team to focus on delivering this item.

See Chapter 5, anti-pattern 25, for a detailed description.

16. Bypassing the Product Owner?

Observation: The stakeholders try to sneak in small tasks by pitching them directly to Developers without going through the Product Owner.

See Chapter 5, anti-pattern 26, for a detailed description.

17. Flow Disruption

Observation: The Scrum Master is unable to prevent stakeholders from disrupting the flow of the work of the Scrum Team during the Sprint.

See Chapter 1, anti-pattern 10, for a detailed description.

PRODUCT BACKLOG AND REFINEMENT ANTI-PATTERNS

These stakeholder anti-patterns result from ignoring the role of the Product Owner, turning them into a scribe. There are two important anti-patterns of this kind.

18. Copy & Paste Product Owner

Observation: The Product Owner creates Product Backlog items by breaking down requirement documents received from stakeholders into smaller chunks.

See Chapter 2, anti-pattern 2, for a detailed description.

19. Prioritization by Proxy

Observation: A single stakeholder or a committee of stakeholders decides on the nature of the Product Backlog. The Product Owner liaises for this or these individuals, passing their ideas to the rest of the team.

See Chapter 10, anti-pattern 1, for a detailed description.

THE DAILY SCRUM

Most anti-patterns in this category result from perceived information needs—think of them as withdrawal symptoms.

20. The Daily Status Report

Observation: The Daily Scrum is a daily status report meeting, and Developers take turns to "report" progress to the Scrum Master, the Product Owner, or maybe even a stakeholder.

See Chapter 3, anti-pattern 16, for a detailed description.

21. Command & Control by Line Managers

Observation: Line managers attend the Daily Scrum to gather performance data on individual team members for later evaluation. Or they wait until the Daily Scrum is over, then reach out to team members directly to obtain status information.

See Chapter 7, anti-pattern 17, for a detailed description.

22. Talkative Stakeholders

Observation: Stakeholders actively participate in the Daily Scrum despite their guest role.

See Chapter 7, anti-pattern 18, for a detailed description.

23. Communicating via Body Language

Observation: Formally, stakeholders remain silent. However, they participate in the Daily Scrum via their body language.

See Chapter 7, anti-pattern 19, for a detailed description.

SPRINT PLANNING ANTI-PATTERNS OF STAKEHOLDERS

24. Forecast Imposed

Observation: The Sprint forecast is not a team-based decision but is driven by individuals. Alternatively, the forecast is not free from outside influence, but stakeholders or the management impose it.

See Chapter 6, anti-pattern 15, for a detailed description.

THE SPRINT REVIEW

This category is often a combination of ignorance, fighting a perceived loss of control, or pulling rank to override Scrum principles.

25. Sprint Review as Approval Gate

Observation: The Sprint Review is used as an approval gate for releasing products to customers.

See Chapter 8, anti-pattern 11, for a detailed description.

26. Internal Stakeholders Do Not Attend

Observation: The Scrum Team runs the Sprint Review for themselves; no internal stakeholders are present.

See Chapter 8, anti-pattern 12, for a detailed description.

27. No Customers Present

Observation: External stakeholders—also known as customers—do not attend the Sprint Review.

See Chapter 8, anti-pattern 13, for a detailed description.

28. Starting Over Again

Observation: There is no continuity in the attendance of stakeholders.

See Chapter 8, anti-pattern 14, for a detailed description.

29. Passive Stakeholders

Observation: The stakeholders are passive and unengaged;

See Chapter 8, anti-pattern 15, for a detailed description.

THE SPRINT RETROSPECTIVE

30. Excluding Stakeholders All the Time

Observation: The Scrum Team categorically rejects having Retrospectives with their stakeholders.

See Chapter 9, anti-pattern 5, for a detailed description.

31. Line Managers Attend

Observation: Line managers regularly participate actively in Scrum Team–internal Retrospectives, raising issues and pointing to possible solutions.

See Chapter 9, anti-pattern 17, for a detailed description.

FOOD FOR THOUGHT

Happy agile islands: An organization strives to become agile, bringing business agility to its core. However, what if that organization fails to understand the necessity of going all in to make this work? What if the transformation process only targets the product or technology part of the organization while the rest pursues business as usual? Can happy agile islands thrive when surrounded by a traditional command-and-control management mindset?

CONCLUSION

There are many different reasons why Scrum stakeholders do not act in line with agile principles. Some result from organizational debt, particularly in legacy organizations from the industrial area. Some are intrinsically motivated, for example, by personal agendas, while others originate from a lack of training or anxieties. Whatever the reason, Scrum stakeholder anti-patterns need to be overcome to achieve organizational agility.

Toolbox

Regarding the following tips and exercises, you will find a detailed description in Appendix B:

- **"Agile Metrics—The Good, the Bad, and the Ugly"** points to suitable agile metrics reflecting either a Scrum Team's progress in becoming agile or an organization's progress in becoming a learning organization.
- **"Building Communities of Practice"** helps win hearts and minds within the organization as it provides authenticity to the agile transformation—signaling that the effort is not merely another management fad.
- **"Sprint Review with Distributed Teams"** describes a string of Liberating Structures microstructures that distributed Scrum Teams can use to organize meaningful Sprint Review events.
- **"Stakeholder Communication Tactics"** comprise eleven proven stakeholder communications tactics to help you not just do agile but become agile.

5 Sprint Anti-Patterns

Introduction

This chapter covers the three Scrum accountabilities (formerly called roles) and addresses interferences of stakeholders and the IT/line management during the Sprint.

Sprints are Scrum's core event, providing a timeboxed frame to deliver a usable Increment. They promote team alignment toward a shared goal, offer frequent inspection and adaptation opportunities, and empower Scrum Teams to adjust work based on feedback, encapsulating key agile principles; see Figure 5.1.

The Purpose of the Sprint

The purpose of the Sprint is clearly described in the Scrum Guide:

- Sprints transform hypotheses into value
- They are consistent, fixed-length events of one month or less, with new Sprints starting immediately after the previous one ends
- Every activity required to achieve the Product Goal, such as Sprint Planning, Daily Scrums, Sprint Review, and Sprint Retrospective, occurs within Sprints
- Sprints foster predictability by inspecting and adapting progress toward a Product Goal at least every calendar month

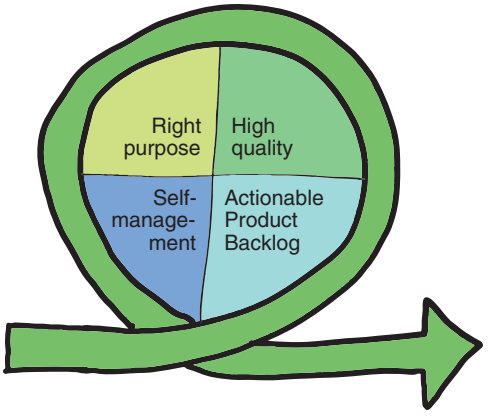

Figure 5.1 Success principles for Scrum in general and Sprints in particular. Use Scrum for the proper purpose: solving complex, adaptive problems, continuously investing in high product quality, creating an actionable Product Backlog,[1] and respecting self-management.

- If a Sprint's length is too long, the market may invalidate a Sprint Goal; complexity and risk may increase
- Shorter Sprints promote more learning cycles and limit the risk of cost and effort to a briefer timeframe, resembling short projects[2]

Scrum as a framework is mainly tactical: the purpose of the Sprint is to deliver value to customers, guided by the Sprint Goal, based on previously explored and validated hypotheses. It does so by releasing Increments and gathering feedback, thus closing the learning loop and starting another round of inspection and adaption.

In pursuit of the Sprint Goal, a Scrum Team also spends time on product discovery, aligns with stakeholders, and refines the Product Backlog.

1. An actionable Product Backlog results from the Scrum Team's continual effort to better understand which ideas may be valuable to the customers—and refining those validated hypotheses into Product Backlog items, ready to be undertaken in a Sprint. An actionable Product Backlog helps the Scrum Team to run an effective Sprint Planning without additional preparation.
2. Ken Schwaber and Jeff Sutherland, "The Sprint," The 2020 Scrum Guide, 2020, https://scrumguides.org/scrum-guide.html#the-sprint.

Sprint Anti-Patterns

Sprint Anti-Patterns of the Product Owner

1. Absent Product Owner

Observation: The Product Owner is absent for most of the Sprint and is not available to answer Developers' questions regarding Product Backlog items from the Sprint Backlog.

See Chapter 2, anti-pattern 20, for a detailed description.

2. Product Owner Changes Work Items from the Sprint Backlog during the Sprint

Observation: The Product Owner cannot let go of Product Backlog items once they become part of the Sprint Backlog. For example, the Product Owner increases the scope of a work item or changes acceptance criteria once the Developers select this Product Backlog item for the Sprint Backlog.

See Chapter 11, anti-pattern 14, for a detailed description.

3. Delaying Product Owner

Observation: Despite the Developers inquiring, the Product Owner does not provide feedback on Product Backlog items once they are done. Instead, they wait until the end of the Sprint to review all done work.

See Chapter 2, anti-pattern 22, for a detailed description.

4. Sprint Stuffing

Observation: The Developers accomplished the Sprint Goal early, and the Product Owner is pushing them hard to accept new work from the Product Backlog to fill the "void" of the Sprint Backlog.

See Chapter 11, anti-pattern 15, for a detailed description.

5. Sprint Cancellations without Consultation

Observation: The Product Owner deems the Sprint Goal obsolete and cancels the Sprint without consulting the Scrum Team.

See Chapter 2, anti-pattern 24, for a detailed description.

6. No Sprint Cancellation

Observation: The Product Owner does not cancel a Sprint whose Sprint Goal can no longer be achieved or has become obsolete.

See Chapter 2, anti-pattern 25, for a detailed description.

SPRINT ANTI-PATTERNS OF THE DEVELOPERS

7. No WiP Limit

Observation: The Developers do not choose to employ a work-in-progress limit to streamline the flow of work.

See Chapter 3, anti-pattern 1, for a detailed description.

8. Lack of Professionalism

Observation: The Developers selectively pull work from the Sprint Backlog based on their personal interests, not on what the Scrum Team needs to get done next. They focus on the creative part and procrastinate in accomplishing the more tedious parts of the work. Consequently, you notice tickets queueing in the code-review column.

See Chapter 3, anti-pattern 2, for a detailed description.

9. Board Out of Date

Observation: The Developers do not update tickets on the Sprint board in time to reflect current statuses, visualizing progress toward the Sprint Goal.

See Chapter 3, anti-pattern 3, for a detailed description.

10. Side Gigs

Observation: The Developers worked on issues outside the Sprint Goal, and the Product Owner learned about those for the first time during the Sprint Review.

See Chapter 3, anti-pattern 4, for a detailed description.

11. Gold-Plating

Observation: The Developers increase the scope of the Sprint beyond the Sprint Goal by adding unnecessary work to the Product Backlog items of the Sprint Backlog.

See Chapter 3, anti-pattern 5, for a detailed description.

Sprint Anti-Patterns of the Scrum Master

12. Flow Disruption

Observation: The Scrum Master is unable to prevent stakeholders from disrupting the flow of the work of the Scrum Team during the Sprint.

See Chapter 1 , anti-pattern 10, for a detailed description.

13. Not Supporting Struggling Developers

Observation: A Developer experiences difficulties accomplishing an issue over several consecutive days, and nobody on the Scrum Team offers help. Moreover, the Scrum Master fails to facilitate the necessary discussion.

See Chapter 1, anti-pattern 12, for a detailed description.

14. #NoRetro

Observation: There is no Retrospective, as the team believes there is nothing to improve, and the Scrum Master accepts this notion.

See Chapter 9, anti-pattern 1, for a detailed description.

Sprint Anti-Patterns of the Scrum Team

15. The Scrum Team Fails to Achieve the Sprint Goal on a Regular Basis

Observation: The Scrum Team misses the Sprint Goal more often than delivering it.

See Chapter 14, anti-pattern 4 for a detailed description.

16. The Maverick and the Sprint Backlog

Observation: Someone adds items to the Sprint Backlog without first consulting the Developers.

See Chapter 11, anti-pattern 3, for a detailed description.

17. The "Hardening" Sprint

Observation: The Scrum Team decides to have a "hardening," or cleanup, Sprint to fix bugs and other accumulated technical issues; see Figure 5.2.

Figure 5.2 There is no such thing as a hardening Sprint in Scrum. The Definition of Done turns quality into a collective team responsibility; all Increments are releasable to customers without repercussions.

Background: Sprints aim to deliver valuable, potentially releasable, done Increments. For that purpose, the Scrum Team creates a Definition of Done to ensure everyone understands the required quality level for an Increment to be releasable to customers.

Not adhering to the Definition of Done, for example, by declaring tasks "done" despite their having associated buggy code violates a core Scrum principle and often signals poor understanding of or commitment to agile principles on the part of the Scrum Team or the organization.

There are several reasons why this anti-pattern may manifest itself; for example:

- **Pressure for speed:** Developers might have felt pressured to deliver to meet an (arbitrary) deadline, and they decided to cut corners by reducing quality.
- **Siloed teams and skills:** The Scrum Team does not handle its own quality assurance but hands the work off to a functional, non-agile silo within the product delivery organization.
- **Skill gaps:** Inadequate skills or knowledge may prevent the team from achieving the quality standards outlined in the Definition of Done.
- **Lack of understanding:** Misunderstanding or lack of awareness about the Definition of Done may result in an inferior quality standard requiring cleanups at a later stage.

- **Organizational culture:** A non-agile culture may incentivize fast results over sustainable development, promoting disregarding Scrum principles.

Remedy: There is no such thing as a hardening Sprint in Scrum.

For the transitional period, consider the following actions to support your team:

- **Foster open discussion:** Encourage open discussions about product quality and the resulting implications of technical debt, emphasizing sustainable development.[3] Retrospectives provide a good starting point.
- **Review the Definition of Done:** If the current Definition of Done is too challenging for the team, reviewing and adjusting it to a more realistic standard might be a good idea, gradually raising the bar as the team's skill improves. Alternatively, increase the quality bar of the current Definition of Done if it has proven inadequate to support the required quality level.
- **Facilitate skill development:** Identify skill gaps and facilitate training or knowledge-sharing sessions to enhance the team's capabilities to meet the Definition of Done.
- **Encourage transparency:** Promote a culture of openness where team members feel comfortable acknowledging when they cannot meet the Definition of Done, leading to proactive problem-solving. For example, the Daily Scrum provides an opportunity to do so.

Learn more about the importance of the Definition of Done and its related anti-patterns in Chapter 15.

18. Sprint Zero

Observation: The Scrum Team starts a new project by dedicating the first Sprint to creating a comprehensive Product Backlog up front.

Background: Embracing the Sprint Zero reintroduces classic project management through the backdoor. It contradicts the principle of empiricism that underpins Scrum by planning excessively in advance in a complex environment, potentially reducing the learning opportunity each Sprint provides. It perpetuates the bias for

3. Learn more about the topic in Kevin Tate's book: *Sustainable Software Development: An Agile Perspective*" (Addison-Wesley, 2005).

action, implying that Scrum's frequent planning activities are not "real" work, undermining the value of iterative development. This practice also instigates loss aversion, where the team might resist changes due to the fear of losing effort already invested in planning, leading teams to continue in a nonviable direction.

Potential reasons for resorting to this practice are, for example:

- **Familiarity and comfort:** Teams transitioning from traditional waterfall development models might find comfort in a Sprint Zero because it resembles the upfront planning phase they are accustomed to.
- **Complex setup:** In specific scenarios, the project might require a complex setup, including system configurations, procurement, or team allocation, which teams may prefer to handle in a Sprint Zero.
- **Product Backlog creation:** Some teams believe they need a Sprint Zero to create an initial, comprehensive Product Backlog, misunderstanding the iterative nature of Product Backlog creation and refinement.
- **Stakeholder expectations:** Teams might use a Sprint Zero to align with stakeholders' expectations for a detailed plan or roadmap, misapprehending Scrum's adaptive, emergent planning approach.
- **Misunderstanding of Scrum:** Sprint Zero might result from misconceptions or misunderstandings of Scrum principles, particularly regarding the importance of empiricism and learning through doing.

Remedy: Simply pointing to the Scrum Guide and its lack of Sprint Zero as a concept may not be adequate to overcome the issue. However, there are some options to counter the anti-pattern:

- **Short Sprint lengths:** Initially, consider suggesting having short Sprints to ease everyone's mind, shorten the time to feedback, and thus prove the effectiveness of an empirical process.
- **Counteract sunk cost fallacy:** Advocate for valuing working software over an extensively planned Product Backlog, and remind the team that it's better to change direction than continue down an unfruitful path.
- **Support Product Backlog refinement:** Help the Product Owner and the team to refine the Product Backlog continuously. A minimal set of Product Backlog items is sufficient to start the learning process in the first Sprint.

- **Promote value in Sprint Planning:** Make the team understand that planning activities within Sprint are real work and essential to manage complexity and mitigate risks.
- **Educate the Scrum Team and stakeholders:** Reinforce the principles of Scrum, such as empiricism and iterative development, to the team and stakeholders. Emphasize the risk of planning too far ahead in complex environments.
- **Address fear of change:** Help the Scrum Team understand that changes are a natural part of agile product development. Foster a culture where learning from change is valued more than adhering to a plan.

19. Delivering Y instead of X

Observation: The Product Owner believes they are getting X. The Developers are working on Y.

See Chapter 12, anti-pattern 8, for a detailed description.

20. New Kid on the Block

Observation: The Scrum Team welcomed a new team member during the Sprint. Unfortunately, they also forgot to address the issue during Sprint Planning, thus ending up overextended.

Background: While it is acceptable to welcome new teammates during a Sprint, the team needs to account for the resulting onboarding effort during the Sprint Planning and adjust its capacity.

Moreover, a new team member should not be a surprise. However, if the newbie turns out to be a surprise, it is an organizational anti-pattern. Some reasons for this behavior may be, for example:

- **Lack of understanding:** The organization may not fully understand Scrum principles, including the importance of self-managing teams.
- **Command-and-control culture:** Organizational culture might be rooted in traditional command-and-control methods, leading to top-down decisions without team consultation.
- **Filling perceived skill gaps:** The organization may perceive an urgent need to fill a skill gap or increase capacity, overlooking the disruption to the team.

- **Resource view of people:** The organization might see individuals as interchangeable resources rather than members of a self-managing Scrum Team.
- **Inadequate trust:** The organization may lack trust in the team's ability to manage its composition, leading to unilateral decisions.

Remedy: Employ a checklist during planning. If team members change frequently, make preparing for these events an item on the list.

Addressing the organizational anti-pattern of new team members showing up without anyone on the Scrum Team knowing requires more effort by building trust with stakeholders and advocating for the team's inclusion in decisions that affect the team composition. A proven way to accomplish both is to run an experiment to find a suitable candidate for a Scrum Team, including the human talent department, line managers, and respective team members.

21. Variable Sprint Length

Observation: The Scrum Team extends the Sprint length by a few days to meet the Sprint Goal.

Background: Generally, if the Sprint length needs to change, the Scrum Team inspects and adapts its working practices during the Retrospective. However, during a Sprint, and contrary to all other Scrum events, the Sprint timebox is fixed. Changing it in an attempt to meet a Sprint Goal that is out of reach is another way of cooking the agile books to match a goal or metric. Extending the Sprint length is not agile; it is cheating. In addition, it devastates stakeholder inclusion, as Scrum events, such as the Sprint Review, lack a proper cadence, reducing your stakeholders' willingness to collaborate.

Remedy: Stop lying to yourself; terminate the Sprint as planned, and during the Sprint Retrospective, analyze what factors contributed to not achieving the Sprint Goal. Based on the analysis results, the Scrum Team may take measures to reduce the risk of repeating the incident.

SPRINT ANTI-PATTERNS OF THE IT MANAGEMENT

22. All Hands to the Pumps without Scrum

Observation: The management temporarily abandons Scrum in a critical situation.

See Chapter 4, anti-pattern 8, for a detailed description.

23. Reassigning Team Members

Observation: The management moves team members from one Scrum Team to another.

Background: Scrum can live up to its potential only if the Scrum Team members can build trust among each other. The longevity of Scrum Teams is beneficial in building that trust. Moving people between teams, on the contrary, fails to consider the difficulties involved in forming teams and simply regards people as interchangeable human resources.

Also, it ignores the preferred team-building practice that Scrum Team members should select themselves. All members need to be voluntarily on a team. Scrum does not work if team members are pressed into service.

However, it is not an anti-pattern if Scrum Teams decide to exchange teammates temporarily. It is an established practice whereby specialists spread knowledge or mentor colleagues. Also, dynamic reteaming does not constitute an anti-pattern when the members of the involved Scrum Teams decide to do so.[4]

Remedy: How can you support the line manager in overcoming this anti-pattern? There are several angles to it; for example:

- **Offer support:** Encourage managers to act as supporters and enablers of the team rather than as controllers of resources. Advocate servant leadership as a management style.
- **Demonstrate value of stable teams:** Share evidence from within the company or industry that illustrates how stable teams lead to higher productivity, improved quality, and morale.
- **Promote team autonomy:** Advocate for team autonomy and decision-making. Show examples of successful self-organizing teams in the organization.
- **Foster trust:** Help build trust in the team's ability to make informed decisions about their composition, including selecting new members.
- **Provide clear communication channels:** Ensure transparent and open communication channels for discussing team changes. Include Scrum Teams in these discussions.

4. Heidi Helfand, Dynamic Reteaming, https://www.heidihelfand.com/dynamic-reteaming/.

None of those approaches promises overnight success. However, perseverance will likely deliver results in the end.

24. Team Autonomy Undermined

Observation: A manager assigns specific tasks directly to Developers, thus bypassing the Product Owner and ignoring the Developers' self-management prerogative. Alternatively, the manager removes a Developer from a team to work on such a task.

Background: This behavior not only violates core Scrum principles but also indicates that the manager cannot relinquish command-and-control practices. They continue to micromanage subordinates, although a Scrum Team could accomplish the task if left to be truly self-organized.

Besides the usual suspects of trust issues, control orientation, and probably a lack of understanding of the details of Scrum, this behavior may also result from the following:

- **Pressure for results:** Managers may resort to old practices when pressured to deliver fast results.
- **Risk aversion:** The perceived risk associated with allowing a team to self-manage might deter managers who are used to minimizing uncertainties.
- **Organizational culture:** A company that values command and control might encourage managers to disregard Scrum's self-management principle when critical problems arise.
- **Career benefits:** The organization may consider taking action as a substantial exhibit of leadership, recommending these "leaders" for another career step.

Remedy: Consider addressing the manager's behavior that violates Scrum in a constructive and nonconfrontational but effective way; for example:

- **Awareness:** Start by raising awareness about the situation and its impact on the Scrum Team's effectiveness, morale, and the overall process.
- **Feedback:** Provide regular feedback on the negative impact of micromanagement. Use concrete examples and data to support your points.
- **Regular Retrospectives:** Include the manager in Retrospectives to learn first hand from Scrum Team members, emphasizing continuous improvement.

- **Show empathy:** Understand their concerns and provide reassurances about the team's capability. Highlight the trust, creativity, and ownership that emerge from self-organization.
- **Offer mentorship:** Consider mentoring the manager and guiding them through accepting Scrum.

If those approaches prove to be unsuccessful, consider escalating the issue:

- **Involve higher management:** If the behavior persists, involve higher management or human talent department to address the issue. They can enforce policy changes or provide necessary interventions.
- **Change the reward system:** Advocate for a change in the reward system to encourage behaviors that align with agile principles.

As an advocate for agile principles and practices, it's important to remember that change is gradual. Shifting from traditional command-and-control mindsets to Scrum's self-management practices requires patience, persistence, and continuous education. Encourage open dialogue about these changes and the benefits they bring to foster an environment of trust and collaboration.

Sprint Review Anti-Patterns of Stakeholders

25. (Regular) Emergency Work

Observation: Someone in your organization has sold a nonexistent feature or functionality to a customer to close a deal, probably already including delivery dates and contractual penalties for nondelivery. Now, they want the Scrum Team to focus on delivering this item.

Background: Generally, there might be moments when outside intervention in the Scrum process is unfortunate but acceptable; for example:

- **Existential business threat:** The new feature or functionality is critical to business survival, and the stakes are too high to follow the normal process—it's a life-or-death situation.
- **Regulatory or legal compliance:** There is an urgent need to comply with regulatory changes or legal requirements.

- **Critical bug fix or security threat:** If a severe bug or security threat emerges, it may necessitate immediate attention, superseding a Scrum Team's planned work.
- **Significant market opportunity:** A rare market opportunity may justify a temporary deviation from Scrum principles.

After all, we are not paid to practice Scrum but to solve our customers' problems within the given constraints while contributing to the organization's sustainability.

Remedy: Emergencies do occur, but they should not become regular events. If the leadership does not acknowledge the exceptionality of the situation, it may derail using Scrum in the organization.

There are a few ways to mitigate this risk; for example:

- **Post-exception review:** After each exception, hold a Retrospective with all stakeholders to discuss what happened, why the exception was necessary, and how to avoid such situations in the future.
- **Feedback loops:** Ensure open communication and feedback loops between sales, leadership, the Scrum Team, and other stakeholders. Employing stakeholder Retrospectives in a regular cadence is helpful.
- **Transparency:** Use agile metrics to make the impacts of exceptions transparent, such as effects on team morale, performance, product quality, and technical debt.
- **Define clear boundaries:** Establish clear criteria for exceptions. Ensure everyone understands that exceptions should be rare and reserved for critical situations.
- **Improve planning:** The occurrence of regular exceptions may indicate room for improvement in product planning, such as creating Product Goals. Also, inspecting the Product Backlog refinement process may prove helpful.

26. Bypassing the Product Owner

Observation: The stakeholders try to sneak in small tasks by pitching them directly to Developers, without going through the Product Owner.

Background: While it clearly violates Scrum principles, there are more straightforward reasons besides pursuing personal agendas or disrespect to explain this behavior:

- Stakeholders may bypass the Product Owner due to a misunderstanding of Scrum roles and principles, mainly the function of the Product Owner.

- They might perceive direct communication with Developers as a faster route to achieve their goals, offering a sense of control.
- They may not trust the Product Owner's prioritization.
- They may embrace habits from traditional project management approaches.
- They may lack visibility into or access to the Product Backlog.

Remedy: As in other cases, merely pointing to the Scrum Guide and rejecting the stakeholders' communication attempts won't solve the issue. Instead, empathize with their situation and help improve overall communication; maybe the Scrum Team failed the stakeholders in the past in that respect. Here are some practical measures to consider:

- **Guide stakeholders:** Stakeholders should be politely redirected to the Product Owner if they approach Developers directly.
- **Educate stakeholders:** Explain the Scrum framework and its specific roles, emphasizing the Product Owner's role and responsibilities. Regular training sessions or workshops can be beneficial.
- **Improve visibility:** Make the Product Backlog accessible and transparent to all stakeholders, thus mitigating their need to bypass the Product Owner.
- **Build trust:** The Scrum Team should use accomplishing Sprint Goals and the Sprint Review to build trust with stakeholders.
- **Feedback channels:** Establish channels beyond the Sprint Review for stakeholders to provide feedback or voice their concerns, for example, by offering regular office hours or stakeholder Retrospectives.

Food for Thought

Sprint length limitations: Is a month-long Sprint too short for accomplishing something meaningful, for example, in hardware development or machine learning? And if so, what are the consequences?

- Does the limitation imply that Scrum is unsuited for these fields, or can we generously decide to extend the Sprint length to meet our team's needs beyond the definition of the Scrum Guide?
- Would that still be Scrum, and does that question matter in the first place as long as we create valuable, viable, usable, and feasible solutions to customers' problems?

- Dying in beauty? Is there a moment when circumstances become so desperate that abandoning Sprints is a valid option? For example, think of a startup running out of cash, and the only way to survive is to achieve a particular milestone defined by a prospective new investor. Would that be when to abandon Scrum's rigorous process to have a fighting chance?

Conclusion

Although the Sprint itself is just a container for all other Scrum events, there are plenty of Sprint anti-patterns to observe. Many of them are easy to fix by the Scrum Team or the Scrum Master when empathizing with the stakeholders' situation and asking, "How did I contribute to this situation in the past?"

However, other Sprint anti-patterns point at corporate issues requiring more than a Sprint Retrospective to change, as they affect, for example, the organizational design or culture, which is beyond the topic of this book.

6 Sprint Planning Anti-Patterns

Introduction

Sprint Planning is the event where Scrum Teams form and choose hypotheses for enhancing their customers' experiences and create a plan for the work they must complete to create an Increment to test these hypotheses. Learn how to improve its effectiveness by avoiding common anti-patterns, from ignoring technical debt to working on random stuff to optimizing for team member utilization; see Figure 6.1.

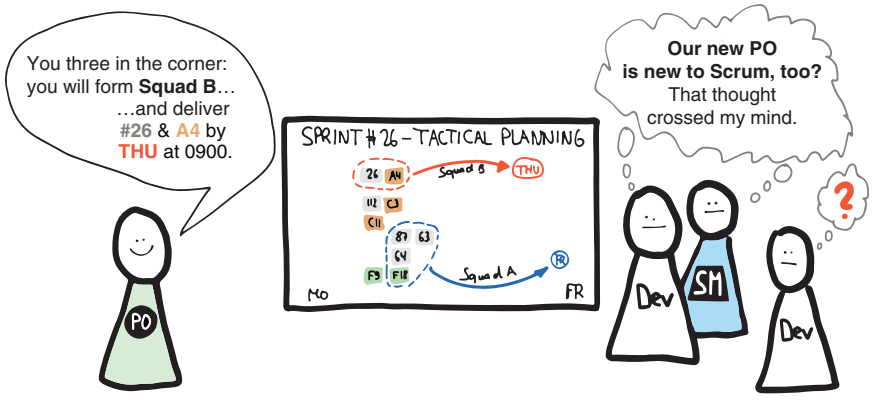
You three in the corner: you will form **Squad B**… …and deliver #26 & A4 by THU at 0900.
PO
SPRINT #26 – TACTICAL PLANNING
Squad B
THU
Squad A
MO
FR
Our new PO is new to Scrum, too? That thought crossed my mind.
Dev
SM
Dev

Figure 6.1 While defining the Product Goal and the business object of the next Sprint are Product Owner responsibilities, creating the Sprint Goal is a collaborative team effort.

Sprint Planning aims to align the Developers and the Product Owner on what to build next to deliver the highest possible value to customers; see Figure 6.2.

- First, the Product Owner points to the team's Product Goal and introduces the desired outcome or business objective of the upcoming Sprint.
- The Scrum Team then collaboratively creates a Sprint Goal to achieve this objective, taking into consideration team member availability.
- Next, the Developers commit to the Sprint Goal and forecast the work required to achieve the Sprint Goal by picking the right items from the Product Backlog and transferring them to the Sprint Backlog. Alternatively, they create additional Product Backlog items.
- The Developers also plan how they will accomplish their forecast.

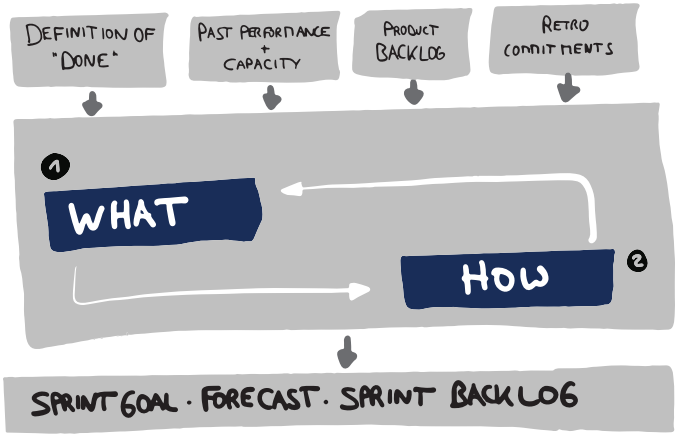

Figure 6.2 The elements of a successful Sprint Planning.

Sprint Planning answers three questions:

1. Why does this particular Sprint matter?
2. What is achievable in this Sprint?
3. How is the selected work accomplished?[1]

1. Ken Schwaber and Jeff Sutherland, "Sprint Planning," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#sprint-planning.

Preparing the Sprint Planning

If the Scrum Team has successfully utilized the Product Backlog refinement process to create and maintain an actionable Product Backlog, Sprint Planning consumes much less time than the eight hours the Scrum Guide lists for a month-long Sprint. In most cases, the Developers and the Product Owner adjust the previously discussed scope of the upcoming Sprint to the available capacity.

Sometimes the Scrum Team will have an actionable Product Backlog, but a valuable new task may arise, and the Product Owner wants to add this task as part of the next Sprint. Consequently, some previously considered Product Backlog items won't make it into the Sprint Backlog. A good Scrum Team can handle that in 10 to 30 minutes before moving on to the decomposition tasks and the initial planning of how the Developers intend to accomplish the work of the Sprint.

Sprint Planning Anti-Patterns

Sprint Planning Anti-Patterns of the Developers

1. No Capacity Check

Observation: The Developers overestimate their available capacity, resulting in a too-ambitious Sprint Goal; see Figure 6.3.

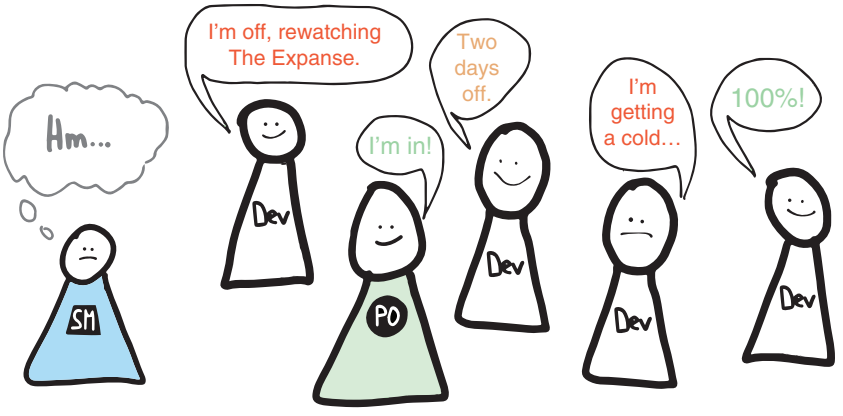

Figure 6.3 Checking your Scrum Team's capacity during Sprint Planning is not "un-agile." On the contrary, it improves forecasting, thus supporting trust-building with stakeholders.

Background: At the beginning of Sprint Planning, the Developers should consider everything affecting their ability to deliver. The list of those issues is long; here are some examples:

- Public holidays
- New members to the Scrum Team requiring onboarding
- Team members quitting, probably demanding a handover
- Team members on vacation or sick leave
- Previous performance
- Corporate overhead, such as all-hands meetings
- Tools, probably in need of acquiring or updating
- Skills required to deliver the new business objective
- Dependencies on other teams or external suppliers
- Technical debt or required maintenance
- Scrum events as practices, such as Product Backlog refinement
- Other events, including team-building exercises, user research during product discovery, and so on

A solid assessment of the situation and existing constraints is critical to choosing an ambitious yet realistic Sprint Goal the team can deliver. Moreover, the regular delivery of valuable Increments is the foundation for building trust with stakeholders and fostering the team's professional reputation within the organization.

Conversely, overcommitment results in undue stress, rushed work, and potential burnout, compromising quality and the ability to meet the Sprint Goal. In addition, it is likely to undermine trust with stakeholders due to the team consistently underdelivering.

Remedy: While a mature team has likely built trust and gained a good understanding of everyone's skills to handle this capacity assessment on the fly, a junior Scrum Team may require some training wheels. A fantastic tool for this purpose is a checklist.

2. What Technical Debt?

Observation: The Product Backlog does not allocate sufficient capacity to maintain the platform, as the Developers do not demand adequate time to tackle technical

debt[2] and bugs to preserve the quality standard of the product. Instead, they fully embrace the feature factory,[3] shipping one new feature after another.

See Chapter 10, anti-pattern 18, for a detailed description.

3. No Slack Time

Observation: The Developers do not demand 20 percent slack time—or unplanned capacity—from the Product Owner.

See Chapter 3, anti-pattern 8, for a detailed description.

4. Planning Too Detailed

Observation: During Sprint Planning, the Developers plan every single task of the upcoming Sprint.

See Chapter 11, anti-pattern 5, for a detailed description.

5. Too Much Estimating

Observation: During the creation of the Sprint plan, the Developers estimate Product Backlog items from the Sprint Backlog down to the level of sub-tasks.

See Chapter 11, anti-pattern 6, for a detailed description.

6. Too Little Planning

Observation: The Developers skip planning altogether and start working immediately.

See Chapter 3, anti-pattern 11, for a detailed description.

---

2. Technical debt is a metaphor in software development describing the future work and complexity resulting from short-term solutions over better, long-term alternatives. Accumulating over time, it can lead to increased maintenance costs, slower development, and lower software quality. Learn more: Martin Fowler, "Technical Debt," May 21, 2019, https://martinfowler.com/bliki/TechnicalDebt.html.
3. John Cutler's *feature factory* concept describes a product development environment where teams focus on delivering features rather than creating value for users and the business. In a feature factory, output (the number of features) precedes outcomes (user experience, the desired change in behavior, accomplishing business objectives, etc.). Unfortunately, this misaligned focus can lead to wasted effort and resources. Scrum practices combat the feature factory by emphasizing collaboration, iterative development, and continuous feedback. Cutler introduced this concept in his article "12 Signs You're Working in a Feature Factory," @johncutlefish's blog, November 17, 2016, https://cutle.fish/blog/12-signs-youre-working-in-a-feature-factory.

7. Team Leads?

Observation: The Developers do not devise a plan to deliver their forecast collaboratively. Instead, a team lead does all the Sprint Planning, and may even assign tasks to individual Developers.

Background: I know senior Developers do not like the idea, but Scrum Teams have no team lead. Scrum is remarkably egalitarian. For example:

- There is no hierarchy within a Scrum Team.
- There are no subroles, either.
- No one can tell anyone what, how, or when to do things.

Of course, as within any social group, Scrum Team members will exhibit deferential behaviors that reflect the informal influence or status of individual team members, despite Scrum's lack of hierarchy. These arise from factors such as tenure, expertise, personality, or perceived value to the team, but they don't confer authority to tell others what to do.

Sometimes, a person (the *team lead*) may try to assert control over a Scrum Team, for reasons such as the following:

- **Comfort zone:** They may be used to a traditional hierarchical management approach and struggle to adapt to the Scrum framework's egalitarian principles. Moreover, they may not trust in the team's ability to self-organize and make sound decisions.
- **Power dynamics:** They may derive status or a sense of power or self-worth from being the decision maker.
- **Perceived efficiency:** They may believe that a single person making decisions is quicker and more efficient without realizing that this can lead to less effective plans and lower team morale.
- **Perception of control:** They may believe that control over decision-making will result in better outcomes, ignoring the value of collective intelligence and shared ownership.
- **Pressure from above:** There may be pressure from stakeholders or upper management to maintain traditional leadership roles, causing the "team lead" to act against Scrum principles.

Unofficial team hierarchy can harm team dynamics, impair decision-making, and kill collaboration. It has no place in Scrum principles and should not be accepted as a reason to replace collaboratively creating the Sprint Plan.

Remedy: Generally, start by emphasizing the importance of everyone's participation; for example:

- **Scrum values:** Reiterate that Scrum is grounded in transparency, inspection, and adaptation. This approach is successful only if everyone actively participates and collaborates in all Scrum events.
- **Diverse input:** Diverse input is crucial to making balanced and effective decisions. Different perspectives can uncover flaws, leading to better problem-solving and decision-making.
- **Empowerment:** Reinforce that Scrum is about self-management. The Developers should collectively decide what they can commit to during the Sprint Planning.
- **Ownership:** Highlight that the more a team member is involved in planning, the more ownership they feel over the work, leading to better quality and commitment.

Moreover, there are some angles for Scrum Masters to address the issue with the *team lead:*

- **Emphasize team success:** Scrum values the team's success over individual accomplishment.
- **Explain role expectations:** Scrum does not recognize the team lead role. Everyone on the team is a Developer with equal authority and responsibility in planning.
- **Highlight benefits of collaboration:** Diverse perspectives and collaborative decision-making often lead to better solutions and a more accurate understanding of the work.
- **Reinforce the concept of self-organization:** Encourage the individual to trust the Scrum Team's ability to manage its work.
- **Address negative impact:** Unilateral decision-making can lead to resentment, lower morale, and reduced productivity within the team.
- **Offer constructive feedback:** Use specific examples to highlight instances where the team lead's approach may have hindered the team's progress or performance.

- **Encourage empathy:** Encourage the individual to empathize with other team members. How would they feel if their input was consistently ignored or undervalued?

When the Scrum Master experiences pressure from the organization for the Scrum Team to have a leader, choosing a suitable course of action becomes more challenging. Of course, the Scrum Master can point to the Scrum Guide's principle that Developers are responsible for creating the Sprint Backlog, emphasizing that this isn't about an individual's control but about leveraging the collective intelligence of the entire team. Moreover, Scrum Masters can facilitate a culture of trust by communicating that mistakes are part of the learning process, supporting the agile value of responding to change over following a plan, as it promotes an adaptive mindset.

SPRINT PLANNING ANTI-PATTERNS OF THE PRODUCT OWNER

8. What Are We Fighting For?

Observation: The Product Owner cannot align the business objective of the upcoming Sprint with the Product Goal and overall product vision, or there is no goal or vision in the first place. This failure likely results in an arbitrary and incoherent assortment of tasks. Consequently, the Scrum Team fails to create a meaningful Sprint Goal.

See Chapter 14, anti-pattern 14, for a detailed description.

9. Unfinished Business

Observation: Unfinished user stories and other tasks from the last Sprint spill over into the new Sprint without any discussion.

Background: While a Scrum Team may think there are good reasons for accepting an occasional spillover—for example, a task's value has not changed—the reality is that the relative importance of that work may have changed, and so the work should be reconsidered when planning the new Sprint. Unfinished Product Backlog items should be moved back to the Product Backlog for further inspection. The rationale is simple:

- If the work items are still valuable, the Scrum Team may choose to include them in the upcoming Sprint.
- If they are no longer valuable or are less valuable than other Product Backlog items, the team will not waste their time working on them but will focus instead

on the items that deliver a higher value during the next Sprint. As a Scrum Team, we want to maximize the value delivered each Sprint; it is not about accomplishing as many work items as possible.

Continuing to work on unfinished Product Backlog items from the previous Sprint should not be automatic; remember the sunk cost fallacy and loss aversion issue.[4]

Remedy: Prior to planning a new Sprint, move all unfinished Product Backlog items back into the Product Backlog. At the next Sprint Planning, consider them with all the other items in your team's Product Backlog.

If a Scrum Team regularly fails to deliver 30 to 40 percent of their original forecast during the Sprint, they may have created a kind of timeboxed Kanban. In this case, now may be the right moment to ask the Scrum Team whether moving to Kanban might suit them better than continuing to use Scrum. If the team considers Scrum still to be its choice, I would recommend putting more energy into Product Backlog refinement and creating meaningful Sprint Goals.[5]

10. Last Minute Changes

Observation: The Product Owner tries to squeeze in some last-minute Product Backlog items that have not been refined.

Background: Principally, it is the prerogative of the Product Owner to make such changes to ensure that the Developers work only on the most valuable tasks at any given time. However, if the Scrum Team is otherwise regularly refining their Product Backlog, these occurrences should be a rare exception. When they happen frequently, there may be problems with the Product Backlog management process; for example:

- The Product Owner may need help managing the Product Backlog.
- The Product Owner needs support to deal with pushy stakeholders who impose their requirements outside the usual Product Backlog process.
- The Product Owner does not communicate well enough with their teammates; typically, this issue results from an inferior Product Backlog refinement process.

4. Wikipedia, "Sunk Cost," updated August 13, 2023, https://en.wikipedia.org/wiki/Sunk_cost#Loss_aversion_and_the_sunk_cost_fallacy.
5. Learn more about Sprint Goals from Maarten Dalmijn, *Driving Value with Sprint Goals: Humble Plans, Exceptional Results* (Pearson, 2023).

Remedy: There are some possibilities to help your Product Owner if they regularly bring unrefined, last-minute Product Backlog items to the Sprint Planning; for example:

- **Facilitate effective product backlog refinement sessions:** Organize regular refinement sessions to involve the Product Owner and the Developers if that is not yet a team practice.
- **Better manage stakeholders:** Coach the Product Owner on effective stakeholder management by teaching them techniques for managing expectations and communicating effectively about the impact of changes on delivery.
- **Improve communication:** Facilitate better communication within the team. If the Product Owner isn't communicating well with their teammates, look into the reasons why and work on solutions. This is an excellent topic for a Retrospective.

For more information on Product Backlog management and the Sprint Review, see Chapters 8 and 10 respectively.

11. Output Focus

Observation: The Product Owner pushes the Developers to take on more tasks than they can realistically handle. The Product Owner may express this in terms of team metrics, such as velocity, to support their desire.

See Chapter 2, anti-pattern 19, for a detailed description.

SPRINT PLANNING ANTI-PATTERNS OF THE SCRUM TEAM

12. Irregular Sprint Lengths

Observation: The Scrum Team has irregular Sprint lengths, adapted to achieve a particular set of Product Backlog items or a specific Sprint Goal.

Background: The Sprint length is not static but adjustable to offer the Scrum Team the best balance between being up to date with customer needs and having sufficient, uninterrupted time to focus on meaningful work. When the team is less familiar with the problem and solution space, Sprints are short. However, Sprint lengths might expand once the Scrum Team becomes more educated, depending on circumstances and the team members' preferences. Also, extending the Sprint length at the end of the year is common when most of the team members are on holiday. (Alternatively, you could, of course, adjust the Sprint Goal.)

Conversely, changing the Sprint length flexibly as needed is challenging, as it will harm the stakeholders' willingness to collaborate, for example, by attending the Sprint Review. The practice forces stakeholders into accepting a scheduling overhead, probably making their attendance at Sprint Reviews less frequent compared to a regular cadence.

Remedy: Instead of changing the Sprint length to accommodate the Sprint Goal, the Scrum Team should invest more effort into appropriately sizing Product Backlog items. In addition, it is advisable to align Sprint length changes with progress on the product side, as the Scrum Team becomes more familiar with its problem and solution space.

Alternatively, the Scrum Team may also gauge their skillset and teamwork to select a manageable Sprint length and consistently stick to it, sizing their Product Backlog items accordingly.

13. Planning Decisions Are Made by Definition of Ready

Observation: The Scrum Team decides that a Definition of Ready is advantageous but handles it dogmatically, blocking any Product Backlog item that doesn't meet the definition from consideration in Sprint Planning.

Background: Typically, a Scrum Team recognizes a Product Backlog item as ready for selection during Sprint Planning when they refine the item to the point that they can turn it into a done Increment within a Sprint. Consequently, accepting an unrefined Product Backlog item into Sprint is a critical topic for team members to discuss. For example, should a valuable user story be postponed to another Sprint just because the front-end designs will not be available for another two working days? Or is this a step toward risking accomplishing the Sprint Goal, as the team does not understand the challenges with this Product Backlog item?

Remedy: My suggestion is simple: take it to the team. If the Developers agree with the circumstances and accept the corresponding Product Backlog item into the Sprint—that is fine.

Find more on the Definition of Ready in Chapter 15, anti-pattern 3. Mike Cohn, a founding member of the Agile Alliance and the Scrum Alliance, also offers insights into the Definition of Ready.[6]

6. Mike Cohn, "Definition of Ready: What It Is and Why It's Dangerous," May 23, 2023, https://www.mountaingoatsoftware.com/blog/the-dangers-of-a-definition-of-ready.

14. No Standard for "Ready"

Observation: The Scrum Team does not have a shared understanding of what a Sprint-ready Product Backlog item requires. Consequently, they select Product Backlog items with a range of readiness levels into the Sprint Backlog.

Background: This issue is the opposite of being dogmatic about applying a Definition of Ready; see preceding anti-pattern. When Developers select unready work items during Sprint Planning, the extra work they must undertake during the Sprint to deal with them creates unnecessary disruptions that may result in failing to achieve the Sprint Goal.

Here are some reasons for differences in understanding what *ready* means:

- **Lack of clear criteria:** Without a shared set of criteria agreed upon by the Scrum Team, Developers may have different perspectives about what constitutes a Sprint-ready Product Backlog item.
- **Diverse backgrounds and experiences:** Developers with diverse experiences, backgrounds, and technical skills might interpret readiness differently based on their previous experiences.
- **Lack of clarity:** If Product Backlog items are unclear or incomplete, Developers might interpret the readiness of Product Backlog items differently.
- **Insufficient understanding of business value or priorities:** If Developers do not fully understand the business value or the priorities of the Product Owner, they might have different views on work items' readiness.

Remedy: Avoid adopting a Definition of Ready to address the issue.

A better solution to the temptation of introducing a formal Definition of Ready is to analyze the root cause of the problem in the Sprint Retrospective. Here are some questions to ask your teammates:

- Do we have a shared understanding of what constitutes a Sprint-ready Product Backlog item? If not, how can we create one?
- Are our past experiences or backgrounds influencing our view of what constitutes readiness? How can we align these different perspectives?
- Are we satisfied with our backlog refinement sessions? How can we make them more effective?

- Do we find Product Backlog items clear and complete? If not, what can we do to improve them? (My tip: Write them collaboratively.)
- Do we all understand the business value and priorities associated with the Product Backlog items? How can we improve our understanding of these aspects?

15. Forecast Imposed

Observation: The Sprint forecast is not a team-based decision but is driven by individuals. Alternatively, the forecast is not free from outside influence, but stakeholders or the management impose it.

Background: This anti-pattern can manifest itself in several ways, such as the following:

- An assertive Product Owner dominates the Developers by defining the scope of the forecast.
- The "technical lead" of the Developers makes a forecast on behalf of everyone else (see above).
- A stakeholder focused on productivity pushes the team to take on more user stories. ("We need to fill the free capacity.")

No matter the reason, the result violates the Developers' prerogative to forecast their work, undermining one of Scrum's first principles: self-management.

Remedy: There are several angles on how to approach the issue:

- **Educate:** Conduct workshops or training sessions to remind everyone of the core principles of Scrum, emphasizing self-management and the importance of the Developers' role in forecasting.
- **Facilitate:** Ensure that Sprint Planning meetings are collaborative and involve input from all Developers. Encourage quieter team members to voice their opinions and prevent a single individual from dominating the conversation.
- **Empower:** Empower and support the Developers to push back when they are given unrealistic expectations or pressured into accepting more work than they can handle.
- **Manage:** Work with stakeholders to manage their expectations and help them understand the importance of team decision-making. Also explain that velocity is not a productivity metric.

- **Coach:** Coach the Product Owner to understand their role better, specifically that they are responsible for the Product Backlog but not for dictating the Sprint forecast.
- **Encourage:** Promote Scrum Values to create an environment where the Scrum Team makes decisions, including the Sprint forecast.
- **Escalate:** Bring the issue to the leadership's attention when attempts fail at the team level to rein in the disrespectful behavior of individuals or stakeholders.

> ***Practical Tales:*** *I worked as a Scrum Master at a large automotive supplier for a new cloud application where worldwide traffic data was "traded." The firm employed about 200 people. The product managers, sales managers, and software architects of this application had the habit of agreeing on feature sets for the upcoming quarters without first consulting with the Scrum Teams. Also, the Product Owners of the teams had no say regarding the Product Goals; their job was to maximize the output.*
>
> *Regularly, the stakeholders of the cloud application sold new features to customers, including delivery deadlines, without anyone on the Scrum Teams knowing. When they announced the new targets, the Developers frantically tried to meet the imposed scope and deadlines, resulting in massive technical issues. As a result, the cloud application was riddled with technical debt and bugs merely three years into its existence.*

SPRINT PLANNING ANTI-PATTERNS OF THE SCRUM MASTER

The Product Owner is responsible for the business objective of the upcoming Sprint. They hence need to guide the Scrum Team's effort during Product Backlog refinement to provide an appropriately prepared, actionable Product Backlog before the actual Sprint Planning. At the same time, the Developers are in charge of selecting the necessary Product Backlog items to meet the collaboratively created Sprint Goal.

So, what are Sprint Planning anti-patterns of the Scrum Master, you may ask yourself? Besides not addressing the previous anti-patterns, or acting as a classic project manager, there is one that sticks out:

16. No Improvements

Observation: While the Scrum Team agrees on improvement items during Retrospectives, they never consider them during Sprint Planning.

Background: Before The 2020 Scrum Guide was released, it was mandatory to address an important improvement issue from the previous Sprint Retrospective during every Sprint to help the Scrum Team improve continuously. The latest Scrum Guide no longer requires that; nevertheless, it is still a good idea that helps the Scrum Team continuously improve.

Remedy: If the team fails to support its own decisions from previous Retrospectives, the Scrum Master should address the issue during Sprint Planning, pointing at the discrepancy between wanting to improve as a Scrum Team and not living up to that intention.

Food for Thought

Continuous Sprint Planning: Imagine we establish a Product Backlog refinement system that enables our team to continuously discover valuable Product Backlog items, supporting the current Product Goal. Simultaneously, during the Sprint, we refine those items to the level that we feel comfortable delivering them successfully. Finally, our Product Owner orders the Product Backlog so that the Scrum Team will maximize the value of their work at any given time. Under these circumstances, do we need a multi-hour planning event at the beginning of a new Sprint Planning? Would a short confirming kick-off be sufficient in most situations?

Is a Product Backlog, per se, wasteful? Some lean practitioners consider the Product Backlog as an artifact already wasteful. They would prefer the just-in-time delivery of valuable Product Backlog items at the beginning of the Sprint Planning.[7]

7. Allan Kelly, "Backlogs, #NoBacklog(s) and Comfort Blankets," September 2022, https://www.allankelly.net/archives/6541/backlogs-nobacklogs-and-comfort-blankets.

CONCLUSION

The Sprint Planning is a core event, defining how your customers' lives will improve with the next Product Increment. It should not be taken lightly. Fortunately, most of the aforementioned Sprint Planning anti-patterns are simple to fix. Just take them to the team and respect Scrum Values, self-management, and Scrum's built-in checks and balances.

Toolbox

Regarding the following tips and exercises, you will find detailed descriptions in Appendix B:

- **"Agile Metrics—The Good, the Bad, and the Ugly"** points to suitable agile metrics reflecting either a Scrum Team's progress in becoming agile or an organization's progress in becoming a learning organization.
- **"Sprint Planning with Distributed Teams"** describes a set of Liberating Structures microstructures that distributed Scrum Teams can use to organize meaningful Sprint Planning events.

7 Daily Scrum Anti-Patterns

Introduction

The Daily Scrum is a critical event for inspection and adaption, run by the Developers, to guide them for the next 24 hours toward achieving the Sprint Goal. The Daily Scrum has the shortest planning horizon in Scrum to help the team deal with the immediate challenges they face.

The Daily Scrum is also a Scrum event that is exceptionally prone to anti-patterns (see Figure 7.1), ranging from becoming a reporting session to assignments to answering three outdated questions[1] that were never intended to represent a Daily Scrum template.

1. The questions appeared in Ken Schwaber and Jeff Sutherland's 2017 Scrum Guide:
   1. What did I do yesterday that helped the Development Team meet the Sprint Goal?
   2. What will I do today to help the Development Team meet the Sprint Goal?
   3. Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal? https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf.

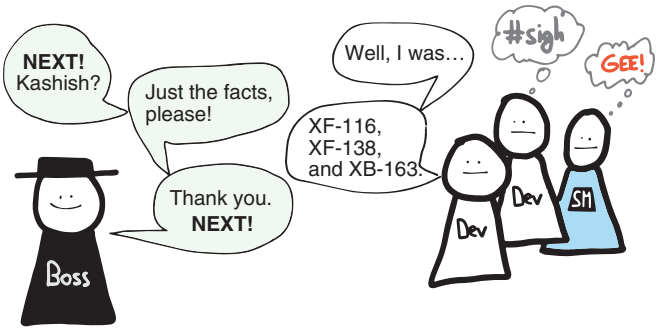

Figure 7.1 The Daily Scrum is not a status reporting session, especially not one driven by stakeholders.

Contrary to popular belief, its 15-minute timebox is not intended to solve all the issues addressed during the Daily Scrum. Instead, its purpose is to create transparency that triggers deeper inspection. If the Scrum Team needs to adapt their plan or their Sprint Backlog,[2] the Developers may handle the resulting issues at any time. In my experience, most Daily Scrum anti-patterns result from misunderstanding this important insight.

The Purpose of the Daily Scrum According to the Scrum Guide

The purpose of the Daily Scrum is clearly described in the Scrum Guide:[3]

- To review progress toward the Sprint Goal and make necessary adjustments to the Sprint Backlog
- To focus on progress and result in an actionable plan for the next day

These important qualifications apply:

- Developers can adjust their plan anytime, not only during the Daily Scrum.
- Developers choose the structure and techniques for the meeting.
- More detailed discussions about adapting or replanning work can occur throughout the day.

2. Technically, the Sprint Backlog comprises the Sprint Goal, the Product Backlog items the Developers deem necessary to accomplish the Sprint Goal, and a plan of how to do so. For this chapter, when I refer to the Sprint Backlog, I refer to its Product Backlog items collection.
3. Ken Schwaber and Jeff Sutherland, "Daily Scrum," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#daily-scrum.

The formality of the Daily Scrum fosters communication by dedicating a short, focused timebox to identify roadblocks and promote quick decisions. The Daily Scrum is not the only time team members talk; they continue to collaborate throughout the day to resolve issues and work toward the Sprint Goal.

DAILY SCRUM ANTI-PATTERNS

An experienced Scrum Team typically won't need more than 10 to 15 minutes to inspect its progress toward the Sprint Goal. Despite this short duration, inexperienced Scrum Teams are able to express a wide variety of anti-patterns. These anti-patterns span a broad spectrum, ranging from behaviors driven by dysfunctional Scrum Teams to apparent failures at an organizational level.

DAILY SCRUM ANTI-PATTERNS OF THE SCRUM TEAM

1. The Lost Scrum Team

Observation: The Developers don't know whether they make progress toward accomplishing the Sprint Goal.

See Chapter 3, anti-pattern 14, for a detailed description.

2. The Daily Status Report

Observation: The Daily Scrum is a daily status report meeting, and Developers take turns to "report" progress to the Scrum Master, the Product Owner, or maybe even a stakeholder.

See Chapter 3, anti-pattern 16, for a detailed description.

3. No Routine

Observation: The Daily Scrum does not happen at the same time and in the same place every day, or even at all.

Background: While routine can ruin a Sprint Retrospective (see Chapter 9), it is important for the Daily Scrum. Think of it as a spontaneous drill: don't put too much thought into the Daily Scrum—just do it.

Remedy: In the Daily Scrum context, routine reduces the Developers' cognitive load, helping them to focus on the Sprint Goal and not their meeting schedule.

Moreover, skipping Daily Scrums establishes bad habits: if you skip a Daily Scrum or two, why not just have them when you need them?[4] Because you cannot know whether you need one until you actually hold it. The intent of the Daily Scrum is to expose issues that would otherwise remain hidden until problems so large they cannot be ignored emerge.

4. Disrespecting Fellow Team Members

Observation: Other team members are talking while someone shares their progress, or they are late to the Daily Scrum, or they don't attend at all.

Background: Developers voluntarily participate in the Daily Scrum because they see that it helps them make progress toward the Scrum Team's goals. If they decide to participate in the Daily Scrum, they need to do so fully, without distraction.[5] Talking over team members while they share their progress and challenges

- Impairs the ability of the team to gather important information,
- Reduces the effectiveness of the team in responding to information, and
- Damages working relationships between team members due to lack of respect.

Remedy: Fostering collaboration among people who strive for independence is tricky: not all team members may find value in the Daily Scrum. Perhaps they have already said everything they have to say and feel the Daily Scrum is a dogmatic exercise. To win these people over, make the Daily Scrum worth everyone's time. If participants of the Daily Scrum have more important things to discuss, why not try to understand what is more valuable to them?

On the other hand, I'm not a fan of a popular coaching tool some people use to bring order to the Daily Scrum: talking tokens. I think that the technique is useless, condescending, and patronizing. It violates the Developer's prerogative of organizing the Daily Scrum.

5. Excessive Feedback

Observation: Team members criticize other team members immediately after their contribution, sparking an argument instead of taking their critique outside the Daily Scrum.

---

4. I would not consider an asynchronous Daily Scrum as "skipping it."
5. Richard Kasperowski, "The Core Protocols v. 3.1," https://thecoreprotocols.org.

Background: Let us consider what the Daily Scrum is not:

- It is not a status report meeting.
- It is not a problem-solving meeting.
- It is also not a place for dealing with conflict.

Its sole purpose is to understand whether a course correction is necessary to stay on track to meet the Sprint Goal.

Some reasons for the observed behavior may include the following:

- **Immediate relevance:** A critic might believe their issue is immediately relevant to the day's work or the Sprint Goal.
- **Frustration or stress:** High pressure or frustration might lead Developers to express criticisms to vent their stress.
- **Avoidance of responsibility:** Some Developers might use criticism to shift blame or avoid responsibility for their tasks or mistakes.
- **Lack of facilitation skills:** The Scrum Master or the facilitating Developer may lack the skills to steer the conversation away from criticisms and toward constructive discussion.
- **Lack of trust:** Developers may decide to publicly point out errors or criticize others to protect their interests.
- **Inefficient feedback culture:** The team's culture may not promote constructive feedback, for example, during Retrospectives.

Remedy: The Daily Scrum is not the place to level criticism. One-on-one sessions with the involved team members are far more effective ways to deal with conflicts and disagreements. If the conflicts arise from the team's way of working, Sprint Retrospectives can also help. General coaching approaches can also be useful:

- **Conflict counseling:** In the case of a deep-rooted conflict within the Scrum Team, consider conflict counseling with a professional facilitator.
- **Coaching facilitation skills:** The Scrum Master can coach the team members facilitating the Daily Scrum to manage the conversation better.
- **Encouraging accountability:** The Scrum Master can discourage blame-shifting and criticism by fostering an environment of ownership and accountability; Scrum is a team sport.

- **Role modeling:** The Scrum Master can model how to give constructive feedback, helping to create a more positive feedback culture within the team.

6. Overcrowding

Observation: The Daily Scrum is ineffective due to too many active participants. As a result, the Scrum Team can no longer meet the timebox, and team members become bored as they wait in line to share.

Background: There is a reason why the Scrum Guide recommends limiting the number of team members to ten (already too many, in my eyes): the larger the team, the more time it takes to align everyone on goals and the steps everyone will take to accomplish those goals.

Remedy: In a Retrospective, discuss splitting the large Scrum Team into two or more smaller ones. If you can't reduce the size of the Scrum Team, try strictly enforcing the timebox. If you have too many issues to cover, try to find other ways of sharing noncritical background information.

DAILY SCRUM ANTI-PATTERNS OF THE DEVELOPERS

7. Unpreparedness

Observation: Developers come unprepared to the Daily Scrum.

See Chapter 3, anti-pattern 15, for a detailed description.

8. Planning Meeting

Observation: The Developers hijack the Daily Scrum to discuss new requirements, refine Product Backlog items, or have a planning meeting, probably collaborating with the Product Owner.

Background: The Daily Scrum is not supposed to provide a time buffer for Product Backlog refinement, as refining new work items is too important to squeeze into the 15-minute timebox of the Daily Scrum. Moreover, the Daily Scrum aims to mitigate the risk of failing to deliver the Sprint Goal. Therefore, while refining Product Backlog items is important, not missing the Sprint Goal is urgent.

Remedy: Instead of misusing the Daily Scrum, Scrum Teams should regularly allocate time for refinement throughout the Sprint. There are multiple benefits of continuously refining Product Backlog items:

- **Flexibility:** Continuous refinement allows the team to adjust and respond to changes or new information as soon as it arises rather than wait for a prescheduled Product Backlog refinement session.
- **Less pressure:** Spreading Product Backlog refinement activities throughout the Sprint reduces the pressure of getting everything perfect in one session.
- **Reduced overhead:** It eliminates the need to organize and conduct a separate meeting, thus reducing overhead and freeing up more time for development work.
- **Better utilization of insights:** The Product Owner and Developers can take immediate action on fresh insights from stakeholders or team members rather than waiting for a formal refinement session.
- **Improved quality:** Continuous refinement can improve the quality of backlog items, as they are revisited and refined multiple times based on ongoing learning and feedback.
- **Reduced risk:** The Product Owner and Developers can identify risks and dependencies sooner, reducing the likelihood of last-minute surprises.
- **Collaboration:** It fosters a more robust collaborative culture as the Product Owner and the Developers work together continuously to refine the Product Backlog.

9. Problem-Solving Session

Observation: During the Daily Scrum, the Developers trigger discussions to solve problems instead of parking those problems to address after the Daily Scrum.

See Chapter 3, anti-pattern 18, for a detailed description.

10. Monologuing

Observation: Team members use the Daily Scrum to monolog about topics outside the scope of the Daily Scrum.

Background: If Developers are direct and concise about their progress and challenges, 60 to 90 seconds per team member should be more than enough time on-air to identify all issues that need the Scrum Team's attention.

Nevertheless, some team members may believe that their contributions benefit the team. Typical reasons for this interpretation may be as follows:

- **Egoism:** Some individuals try to take control of every meeting they attend, often seeking to be the focal point of discussion.
- **Transparency:** They might think discussing broader topics brings more visibility to their work and efforts.
- **Idea sharing:** They might see the Daily Scrum as an opportunity to share new ideas and innovations with the team.
- **Learning opportunity:** They might view these discussions as opportunities to learn from their colleagues' experiences and expertise, benefiting their work.
- **Building rapport:** They might believe that discussing unrelated topics can help build rapport and foster team camaraderie, indirectly contributing to Scrum Team's productivity.
- **Avoiding other meetings:** They may believe that discussing broader topics in the Daily Scrum can reduce the need for additional meetings.
- **Breaking monotony:** They may believe that diversifying the discussion will make the Daily Scrum more engaging and less monotonous.

Consequently, figuring out what motivates your teammates to hijack the Daily Scrum is essential.

Remedy: Before addressing this behavior with the team members, it's crucial to comprehend their reasons for dominating the conversation. Then, through constructive dialogue, you can help them understand how their extensive contributions might undermine the purpose of the Daily Scrum, and encourage them to be more succinct in their responses.

If that approach does not work, consider some of the following techniques:

- **Lead by example:** Demonstrate effective communication during the Daily Scrum by keeping your updates and comments brief, focused, and relevant.
- **Radical candor:** Gently interrupt off-topic discussions and redirect the conversation back to inspecting progress toward the Sprint Goal, reinforcing the purpose of the Daily Scrum.

- **Promote a parking lot:** If off-topic issues arise, suggest parking them for discussion outside the Daily Scrum; see preceding anti-pattern Problem-Solving Session.
- **Encourage peer feedback:** Facilitate a feedback session where team members can share their feelings about the lengthy discussions in a respectful, constructive manner.

Also, there are two more controversial techniques:

- **Implement time limits:** Allocate a specific time limit for each member's update and track it with a time-keeping device.
- **Use a talking token:** Introduce a talking token that the speaking person must hold. This approach establishes a speaking routine and can visually remind members to be concise and stay on topic.

Personally, I would not apply time limits or talking tokens. While the techniques may seem effective in controlling the duration of the Daily Scrum, they can inadvertently stifle communication and collaboration, which are at the heart of agile principles:

- Strict time limits might prevent team members from sharing important information or raise potential blockers out of fear of overrunning their allocated time. As a result, team members could fail to address critical issues, reducing the effectiveness of the Daily Scrum.
- Using talking tokens might make the meeting feel overly regimented, which can stifle open and natural discussion. It may also create an unnecessary focus on the token and the routine rather than on the content of the discussion. While being concise is important, fostering a collaborative and safe environment is critical in Scrum.

Finally, take it to the team's Retrospective to discuss the issue and find a solution, for example, by amending the working agreement accordingly.

11. No Impediments

Observation: No Developer reports any obstacles; no one needs help or support from their teammates.

See Chapter 3, anti-pattern 22, for a detailed description.

DAILY SCRUM ANTI-PATTERNS OF THE PRODUCT OWNER

12. Assignments

Observation: The Product Owner assigns tasks directly to team members during the Daily Scrum.

See Chapter 2, anti-pattern 26, for a detailed description.

13. Introducing Additional Work

Observation: Despite good Product Backlog refinement and Sprint Planning, the Product Owner or even stakeholders attempt to introduce new work to the current Sprint during the Daily Scrum.

See Chapter 2, anti-pattern 27, for a detailed description.

DAILY SCRUM ANTI-PATTERNS OF THE SCRUM MASTER

14. Daily Scrum Enforcer

Observation: The Scrum Master is enforcing the Daily Scrum, as the Developers show little motivation to self-manage around the event.

Background: It is the task of the Scrum Master to ensure "that all Scrum events take place and are positive, productive, and kept within the time-box."[6] Consequently, this obligation applies to the Daily Scrum, too.

However, suppose none of the Developers believe that Daily Scrum is a sound investment. In that case, the Scrum Master cannot simply compensate for this lack of intrinsic motivation by using the Scrum Guide as a lever to force movement. Doing so may confirm the Developers' suspicion that Scrum is merely another management methodology to maximize output while paying lip service to agile principles such as self-management. Instead, the Scrum Master needs to help the Developers turn the Daily Scrum into an event that all Developers consider worth their time.

6. Ken Schwaber and Jeff Sutherland, "Scrum Master," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#scrum-master.

Remedy: Some of the following approaches may be helpful to overcome a Scrum Master's enforcer mentality:

- **Trust the team:** Believe in the ability of the Developers to self-organize and manage their work; give them the benefit of the doubt, but observe.
- **Foster autonomy:** Facilitate the development of the team's ability to self-organize and self-manage by teaching, coaching, or just creating the space for this to occur. The Daily Scrum may be a starting point.
- **Self-restraint:** Practice restraint, letting the team handle their responsibilities and stepping in only when necessary.
- **Facilitate, don't direct:** Facilitate discussions and decision-making processes without dictating the outcomes.
- **Lead by example:** Embody the values and principles of Scrum and Agile in all actions and interactions. Living these values creates a role model for the team to follow.

15. Not Supporting Struggling Developers

Observation: A Developer experiences difficulties accomplishing a task over several consecutive days, and nobody on the Scrum Team offers help. Moreover, the Scrum Master fails to facilitate the necessary discussion.

See Chapter 1, anti-pattern 12, for a detailed description.

16. Not Preventing Stakeholders from Attendance

Observation: Although the Developers decided to limit attendance to the Daily Scrum to the Scrum Team members, stakeholders show up uninvited, and the Scrum Master fails to address the issue.

See Chapter 1, anti-pattern 13, for a detailed description.

DAILY SCRUM ANTI-PATTERNS OF THE STAKEHOLDERS

17. Command & Control by Line Managers

Observation: Line managers attend the Daily Scrum to gather performance data on individual team members for later evaluation. Or they wait until the Daily Scrum is over and reach out to team members directly to obtain status information.

Background: This behavior threatens the ability of a Scrum Team to self-manage. In fact, it may be concrete evidence that the Scrum Team is not self-managing, either because it is unable to self-manage or because it is not allowed to self-manage. As a result, the team itself may fall apart and may simply be a group of individual Developers working largely independently. This threatens their ability to achieve the Sprint Goal.

Often, this management behavior is rooted in management practices from the industrial era and based on Taylorism or scientific management.[7]

Remedy: Here are some approaches to convince line managers that their attendance at Daily Scrum events reduces the effectiveness of Scrum Teams:

- **Explain team vs. individual focus:** Emphasize Scrum's focus on team performance rather than individual output. Explain that gathering personal performance data during Daily Scrums may compromise this focus.
- **Demonstrate consequences:** Show how their presence can lead to distorted communication, where Developers prioritize looking good individually over an open discussion about progress and impediments.
- **Highlight psychological safety:** Explain that constant observation may create a pressure-filled environment that prevents Developers from openly discussing challenges, as they may fear compromising career perspectives.
- **Show impact on Sprint Goal:** Discuss how focusing on individual performance and pleasing the line manager can detract from the team's collective effort toward the Sprint Goal.
- **Promote transparency:** Assure managers that Scrum's transparency will provide them with necessary information about the team's progress, for example, through the Sprint Review and artifacts such as the Product Backlog, the Sprint Backlog, and the Product Increments.

18. Talkative Stakeholders

Observation: Stakeholders actively participate in the Daily Scrum despite their guest role.

7. Wikipedia, "Scientific Management," updated August 19, 2023, https://en.wikipedia.org/wiki/Scientific_management.

Background: When Developers agree on inviting stakeholders to participate in the Daily Scrum, they expect the stakeholders to listen in but not distract from the inspection of progress toward the Sprint Goal. Stakeholders, such as subject matter experts, can actively participate only when prompted by the Developers.

Violating this foundational principle of the Daily Scrum can result in levering out self-management and reintroducing classic line management through the backdoor; see Figure 7.2.

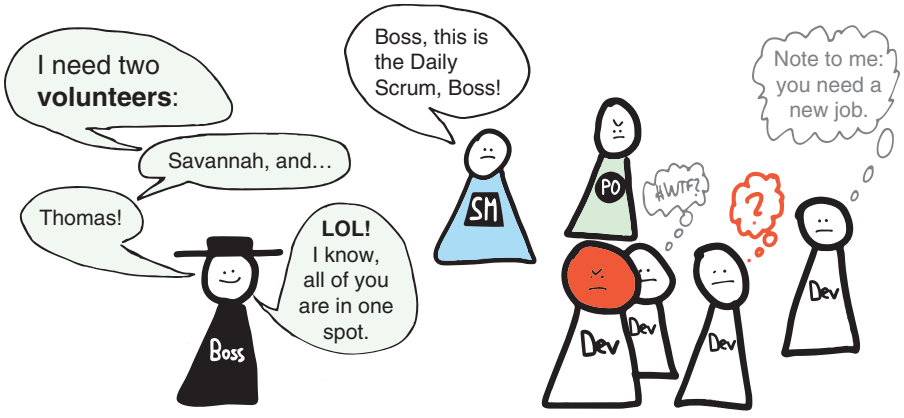

Figure 7.2 Stakeholders shall not actively participate in the Daily Scrum or ignore Scrum's first principle of self-management.

Consequently, Scrum Masters should immediately handle the situation and help the stakeholders understand their role in the Daily Scrum. After all, it is the Developers' prerogative to run the Daily Scrum in a way they see fit. Stakeholders can participate in the Daily Scrum only because the Developers agree to it.

Remedy: Here are ways that Scrum Masters can convince stakeholders to restrain themselves from participating in Daily Scrum events:

- **Clarifying the stakeholder role:** Explain the observer role of stakeholders in the Daily Scrum, emphasizing that only Developers communicate actively.
- **Demonstrating impact:** Show how stakeholder intervention can disrupt the focus of the Daily Scrum and derail the Developers' concentration on their Sprint commitments.

- **Pointing to the Sprint Review:** Highlight the Sprint Review as the appropriate event for stakeholders to engage and provide feedback on the work completed actively.
- **Opening communication channels:** Arrange separate meetings where stakeholders can discuss their concerns or insights with the Scrum Team when the Sprint Review is insufficient to meet the stakeholders' communication needs.

19. Communicating via Body Language

Observation: Formally, stakeholders remain silent. However, they participate in the Daily Scrum via their body language.

Background: Again, stakeholders are supposed to listen in but not distract the Developers. Rolling eyes and face-palming are as powerful as the spoken word. Moreover, even subtle forms of body language represent communication, as one cannot "not communicate." Furthermore, some stakeholders may have a naturally intimidating presence that can harm communication among the Developers. Think of Elon Musk attending your Daily Scrum.

Remedy: The Scrum Master should share their observation with the respective stakeholders, as they may not be aware of their influence. If they do not believe they influence the Daily Scrum with their body language, suggest recording some sessions for later analysis.

Moreover, consider polling the team members anonymously regarding this effect and sharing the information with the stakeholders.

Otherwise, refer to the Remedy section of the preceding Talkative Stakeholders anti-pattern for more suggestions.

FOOD FOR THOUGHT

Synchronous and asynchronous Daily Scrum events: Some teams like to have their Daily Scrum in a communication application of their choice, particularly those who are not co-located. But, of course, using Slack does not manifest an anti-pattern per se; it is the prerogative of the Developers to run the Daily Scrum in any fashion that serves its purpose: inspect the plan for the next 24 hours to meet the Sprint Goal. I was even working with a co-located Scrum Team—sitting around a large table—that

used a synchronous Slack session as their preferred way of having their Daily Scrum. It worked well. But what about an asynchronous Daily Scrum in Slack?

Substituting the Daily Scrum: What about skipping the Daily Scrum altogether, as the Developers are heavily using more or less automated messaging systems on most aspects of their daily work, while they can handle the rest during other events, such as pairing or mobbing sessions? Is it beneficial to allocate 15 minutes every day to the Daily Scrum under these circumstances? Or is that mere dogmatism?

Extending the 15-minute timebox: Suppose a Scrum Team works in an environment where they need to allocate a significant part of the daily capacity to operational issues. While this work is not part of the Sprint Goal, it is essential for the organization's sustainability. Typically, dealing with these issues requires everyone on the team to participate. Should the team address these issues during the Daily Scrum, even at the cost of extending its timebox?

Conclusion

Given the importance of the Daily Scrum for the success of the Scrum Team's effort to achieve the Sprint Goal, its anti-pattern density is no surprise. Unfortunately, many people seem ignorant of (or at least less well educated about) its purpose. Or they—intentionally or not—interfere with the Developers' self-management.

However, the solution to these problems lies mainly with the Scrum Master. They are tasked with educating and guiding all participants to uphold the purpose of the Daily Scrum by fostering a culture of transparency, respect, and shared accountability. It is the only way to make the Daily Scrum a valuable tool in achieving the Sprint Goal.

Toolbox

Regarding the following tips and exercises, you will find detailed descriptions in Appendix B:

- **"Daily Scrum with Distributed Teams"** points to a set of Liberating Structures that support distributed or hybrid teams in organizing a meaningful Daily Scrum.

This page intentionally left blank

Sprint Review Anti-Patterns

Introduction

Are we still on track to accomplish the Product Goal? Answering this question and adapting the Product Backlog in a collaborative effort of the Scrum Team with internal and external stakeholders is the purpose of the Sprint Review; see Figure 8.1.

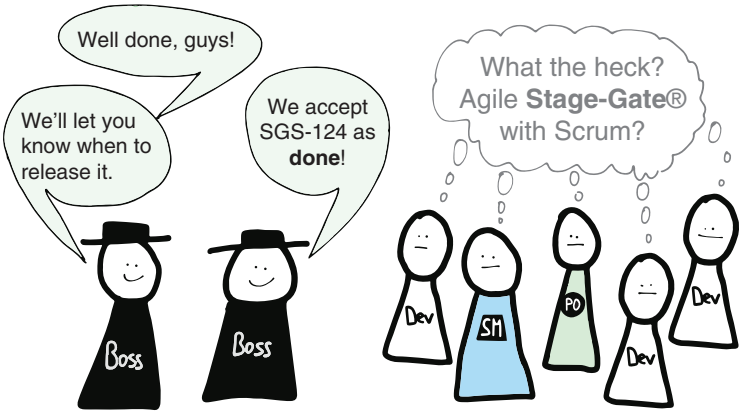

Figure 8.1 The Sprint Review is not an acceptance gate but answers a simple question: Are we still on track to accomplish the Product Goal?

THE SCRUM GUIDE ON THE SPRINT REVIEW

According to the Scrum Guide, the Sprint Review serves the following purpose:[1]

- **Evaluate Sprint's accomplishments:** Assess the outcomes achieved during the Sprint and the value delivered to stakeholders in a complex environment.
- **Reflect on progress toward the Product Goal:** Discuss how the outcomes contribute to achieving the Product Goal and explore whether any adaptations are needed.
- **Plan future adaptations:** Identify opportunities for adaptation and learning to enhance the team's ability to create value and respond to evolving needs.
- **Consider environmental changes:** Understand the impact of any internal or external factors on the team's value creation and consider these when planning adaptations of the Product Backlog or Product Goal.
- **Collaboratively decide on next steps:** The Scrum Team and stakeholders work together to determine the most appropriate adaptations for the upcoming Sprint based on the insights gathered during the Sprint Review.
- **Adapt Product Backlog:** Update the Product Backlog to reflect new insights, stakeholder feedback, or market changes, ensuring it remains focused on delivering value.
- **Encourage collaborative discussions:** Promote active engagement and open conversations among participants—stakeholders and team members alike—fostering a collaborative atmosphere.

The Sprint Review is empiricism at work: inspect the Product Increment(s) and adapt the Product Backlog. It is the best moment to create or reaffirm the shared understanding among all participants whether the Product Backlog still reflects the best use of the Scrum Team's time, thus maximizing the value delivered to customers within the given constraints while contributing to the viability of the organization. It is also because of this context that calling the Sprint Review a "demo" does not match its importance for the effectiveness of the Scrum Team.

Moreover, after the Retrospective, the next Sprint can start as the Scrum Team has come full circle.

1. Ken Schwaber and Jeff Sutherland, "Sprint Review," The 2020 Scrum Guide," 2020, https://scrumguides.org/scrum-guide.html#sprint-review.

Sprint Review Anti-Patterns

Sprint Review Anti-Patterns of the Scrum Team

1. No Sprint Review

Observation: The Sprint Review is canceled because the Developers did not meet the Sprint Goal.

Background: Just because the Scrum Team failed to achieve the Sprint Goal does not mean they have nothing to talk about. A Sprint Review is necessary to create transparency with stakeholders, particularly in such a situation.

There are plenty of reasons why the team may have missed the Sprint Goal; for example:

- An unexpected absence of team members
- Unexpected complexity
- Unexpected technical debt or lack of technical skills
- Unavailability of tools
- Team members quit the team
- An emergency overrode the original Sprint plan
- Dependencies on other teams or outside suppliers
- An overly ambitious, poorly formed, or misinformed Sprint Goal

If the Sprint goes sideways, if the Scrum Team fails to deliver the Sprint Goal, own the problem and get ahead; see Figure 8.2.

Remedy: The Scrum Team plays an infinite game[2] regarding gaining the stakeholders' trust. Consequently, inspect the Increments that the Developers managed to accomplish, share your story, and adapt the Product Backlog. You are most likely still working on the same Product Goal.

2. In his book *The Infinite Game* (Penguin Business, 2020), Simon Sinek argues that in business, the goal isn't to *win* but to keep playing and evolving. Unlike finite games with clear endings and winners, infinite games continue indefinitely with evolving players and rules. Companies should aim to outlast and out-innovate, focusing on long-term sustainability over short-term gains. This approach requires shifting from a fixed to a growth mindset, emphasizing progress and innovation over finite success metrics.

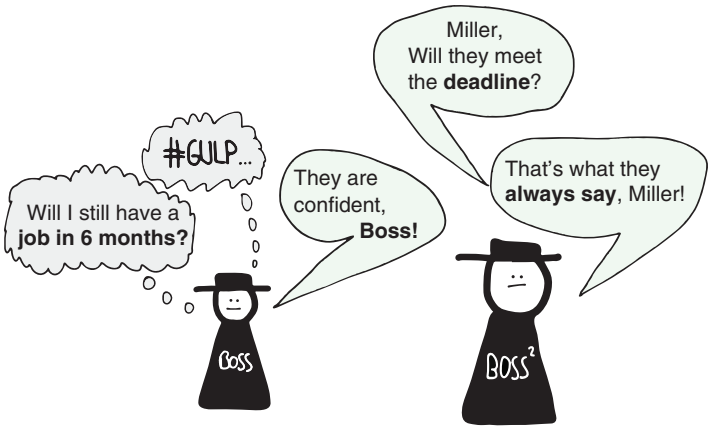

Figure 8.2 Building stakeholder trust is essential for a Scrum Team's success. While it may take weeks or months to do so, it takes little more than a missed delivery or two to set your team back to square one.

2. Ignoring the Purpose of the Sprint Review

Observation: The Scrum Team does not use the Sprint Review to inspect progress toward the Product Goal with the stakeholders.

Background: Creating transparency about the Scrum Team's goals and receiving feedback and input from stakeholders regarding the outcome of the previous Sprint and current developments in the market is the purpose of the Sprint Review.[3] The Sprint Review helps to mitigate the Scrum Team's risk of decoupling itself from the market. Consequently, failing to elicit stakeholder feedback fails to reduce this risk.

Instead, the Scrum Team should embrace the opportunity to learn from knowledgeable peers outside the Scrum Team and adapt its Product Backlog accordingly. Following are some reasons why the Scrum Team may fail to do so:

- **Lack of understanding:** The Scrum Team might not fully comprehend the importance of the Sprint Review in assessing progress toward the Product Goal.
- **Disconnected stakeholders:** Stakeholders might not be adequately involved in the Scrum process, making it challenging to review progress with them.

3. Bernd Schiffer: Sprint Review, a Feedback Gathering Event: 17 Questions and 8 Techniques, 2015-09-15: https://agiletrail.com/2015/09/19/sprint-review-a-feedback-gathering-event-17-questions-and-8-techniques/.

- **Fear of difficult conversations:** The team may want to avoid challenging conversations about progress, blockers, or necessary changes.
- **Lack of clarity:** The Product Goal might not be clearly defined, making it hard to measure progress.

Remedy: Given the importance of stakeholder feedback, the Scrum Team should consider the issue urgent and not postpone understanding the reason for their ineffective Sprint Reviews.

There are several ways to tackle the root causes of the issues:

- **Clearly define the Product Goal:** Make sure the Product Goal is clear, measurable, and understood by everyone on the team and the stakeholders. This transparency makes reviewing and assessing progress during the Sprint Review easier.
- **Educate the Scrum Team and its stakeholders:** Conduct Scrum training sessions to ensure the entire team and its stakeholders understand the importance of the Sprint Review and its role in achieving the Product Goal.
- **Engage stakeholders:** Involve stakeholders more actively in the Scrum process, ensuring they're present at Sprint Reviews and that they understand their roles and the importance of their feedback for the Scrum Team's success.
- **Encourage open communication:** Foster a safe environment for open dialogue about progress, challenges, and changes needed. Remind the team and the stakeholders that these conversations are crucial for improvement.

Generally, I also recommend running a joint Retrospective with your stakeholders to bring the issue and your team's needs to their attention; see the toolbox tip:

Toolbox Tip

"Meta-Retrospectives—How To Get Customers and Stakeholders Onboard" (see Appendix B) describes an excellent exercise to foster collaboration within the extended team, create a shared understanding of the big picture, and immediately create valuable action items.

3. Sprint Task Accounting

Observation: The Developers demonstrate every task accomplished, diluting the impact of their real achievements and boring the stakeholders, causing them to disengage in the future.

Background: Sprints are outcome focused, not output focused. Sprint Reviews should showcase what the team achieved, not the work that they did to achieve it.

Nevertheless, the team members may feel the urge to explain what they did during the Sprint, particularly in organizations whose culture still values output over outcome. Here are some reasons the Developers may overexplain their work:

- **Lack of confidence:** The Scrum Team may lack confidence in the value of their Sprint Goal and feel the need to demonstrate every detail to justify their efforts.
- **Misunderstanding the purpose:** An inexperienced Scrum Team may misunderstand the purpose of the Sprint Review and perceive it as a detailed status report rather than an opportunity to inspect the Increment and adapt the Product Backlog.
- **The complete picture:** The team might be afraid that if they don't present all work, stakeholders may miss essential details, influencing future decisions.
- **Miscommunication:** The team could be overcompensating for previous feedback or comments that stakeholders do not understand what the Scrum Team does.

Remedy: Refrain from boring stakeholders by discussing everything that the team did to achieve the Sprint Goal. Stakeholders are not task accountants. My suggestion to overcome the mentality is simple: tell a compelling story at the beginning of the Sprint Review on the goal the Scrum Team ventured out to achieve and engage the stakeholders with that narrative. Leave out those work items that are not relevant to the story.

SPRINT REVIEW ANTI-PATTERNS OF THE PRODUCT OWNER

4. The Selfish Product Owner

Observation: Product Owners present the team's accomplishments as their own to the stakeholders.

See Chapter 2, anti-pattern 28, for a detailed description.

5. "Acceptance" by the Product Owner

Observation: The Product Owner uses the Sprint Review to "accept" Product Backlog items that Developers consider done as they adhere to the team's Definition of Done.

See Chapter 2, anti-pattern 29, for a detailed description.

6. The "Know-It-All" Product Owner Loving Their Ideas

Observation: The Product Owner does not accept feedback from stakeholders or the Developers, stubbornly pursuing their own ideas. When presented with feedback that does not reinforce their preconceptions, the Product Owner will say that the stakeholders "just don't understand."

Background: It's easy for the Product Owner to fall in love with their solution to the exclusion of customer and stakeholder feedback to the contrary.

This attitude is reflected in the many reasons why a Product Owner ignores feedback, comments, and critique from stakeholders and teammates; for example:

- **Lack of trust:** The Product Owner may not trust the feedback and ideas of stakeholders or team members, believing only in their own vision.
- **Misunderstanding of role:** They may misunderstand their role's collaborative and receptive nature, thinking they're solely responsible for defining the product's direction.
- **Ego issues:** Personal pride and ego might prevent them from accepting other perspectives, believing their ideas are superior.
- **Fear of losing control:** They might fear losing control over the product's direction if they consider input from others.
- **Inexperience:** They might be inexperienced in a collaborative, feedback-rich environment like Scrum, leading to a tendency to dismiss others' ideas.
- **Fixed mindset:** They may have a fixed mindset, resisting change and new ideas in favor of their own.
- **Lack of understanding:** They may lack understanding of the problem space or user needs, relying heavily on their personal perspective instead.
- **Pressure:** External pressures, such as deadlines or stakeholder expectations, might push them to stick to their own ideas, dismissing feedback as time-consuming.

- **Poor communication skills:** They may struggle to communicate effectively and engage in constructive dialogue, leading to a dismissal of feedback.

Remedy: Product Owners do not get paid to roll out their solution. This "living in their Product Owner bubble" approach violates the prime purpose of the Sprint Review event: to determine collaboratively whether the Scrum Team is still on track to meeting the Product Goal—which requires openness on the side of the Product Owners in the first place.

Once we understand the challenges of a particular case, we can move on and help our Product Owner to overcome this team-limiting attitude:

- **Encourage collaboration:** Foster an environment that encourages collaboration and open communication. Make sure everyone understands the value of different perspectives and contributions.
- **Role-model openness:** As a Scrum Master, model the behavior you want to see. Show the Product Owner how openness to feedback and ideas can lead to better decision-making and outcomes.
- **Promote empathy:** Encourage the Product Owner to spend time with users and stakeholders to better understand their needs and perspectives.
- **Encourage feedback sessions:** Organize regular feedback sessions where team members can share their thoughts and suggestions openly.
- **Coach on communication:** Offer coaching on effective communication and listening skills, helping the Product Owner to understand and value others' inputs.
- **Provide mentorship:** Find the Product Owner a mentor who has successfully navigated similar challenges.
- **Highlight success stories:** Share examples of how feedback and collaboration have led to successful outcomes in the past.
- **Facilitate discussion:** During Sprint Reviews and other Scrum events, facilitate discussions that allow everyone to share their ideas and feedback.
- **Promote a growth mindset:** Encourage a culture of learning and continuous improvement, emphasizing that everyone's ideas are valuable for the team's growth.

Sprint Review Anti-Patterns of the Developers

7. Death by PowerPoint

Observation: The Developers bore participants of the Sprint Review to death with PowerPoint presentations.

Background: The foundation of a successful Sprint Review is "show, don't tell," or even better: let the stakeholders drive their own discovery. The Sprint Review is not a demo, a sales pitch, or a presentation that carefully avoids anything that doesn't support a narrative of success. Instead, it is an essential opportunity to share what the Scrum Team has accomplished, inspect their achievements and challenges, provide valuable feedback, and adapt the Product Backlog collectively. It is focused on creating transparency on the state of the Scrum Team's progress toward the Product Goal.

Remedy: Here are some suggestions on how to quit PowerPoint and engage all Sprint Review participants:

- **Preparation:** Brief stakeholders about the Sprint's outcome in advance to allow them to come prepared with relevant questions and feedback.
- **Showcase value:** Show how the new Increments deliver value, for example, by sharing customer feedback, metrics, or other evidence of impact.
- **Stakeholders at the helm:** To make the Sprint Review more engaging and to provide more feedback opportunities, let stakeholders try out the new features instead of having the team present them.
- **Focused discussions:** Keep discussions focused on the Increments, progress toward the Product Goal, and how they bring value to users and the business.
- **Facilitate open dialogue:** Encourage open discussion and debate by including everyone and giving everyone a voice. Facilitate a safe environment where stakeholders can express differing opinions.
- **Ask for feedback:** Explicitly ask stakeholders for their feedback and opinions, thus making them feel valued and encouraging active participation.
- **Recognize contributions:** Acknowledge and thank stakeholders for their contributions, feedback, and participation to create a positive atmosphere and encourage future engagement.
- **Keep it brief and concise:** Respect stakeholders' time by keeping the Sprint Review focused and strictly within its timebox.

8. Same Faces Again

Observation: It is always the same Developers who participate in the Sprint Review; however, some Developers never participate.

Background: The limited attendance of Scrum Team members is challenging, as the Sprint Review needs all Scrum Team members to attend so that important perspectives are not ignored. The rationale is that in a complex environment, there are no experts, and the strength of the Scrum Team is rooted in the collaboration of all its members. As a result, everyone can contribute to progressing toward Sprint Goals and, consequently, the Product Goal.

Moreover, the nonparticipating team members miss opportunities to understand the stakeholders' problems better and build rapport and trust with them. In addition, the absence of some team members may damage the team's reputation in the eyes of the stakeholders: "Why don't they respect our commitment by attending the Sprint Review themselves?"

Remedy: The challenge for Scrum Masters in this situation is that you cannot—and should not—enforce your teammates' participation. Instead, make it interesting enough that everyone wants to participate.

If this is not happening, the Scrum Master should ask themself how they may have contributed to this situation in the past. Some possible reasons follow:

- **Lack of support:** The Scrum Master may have failed to aid the Product Owner in facilitating the Sprint Reviews, causing a lack of team interest.
- **Lack of psychological safety:** The Scrum Master failed to foster a safe environment for open communication.
- **The Scrum Team plays the blame game:** The Scrum Master let previous Sprint Reviews turn into blame sessions, which now can deter team members.
- **Lack of conflict resolution:** Ignoring conflicts can push team members to skip Sprint Reviews.
- **Lack of inclusion:** Everyone needs to be included in a Sprint Review, or some may feel unwelcome.
- **Boredom:** The Sprint Review is unappealing (see preceding anti-pattern Death by PowerPoint).

- **Disinterest:** The stakeholders do not engage with the team members during the Sprint Review.

However, there is an event tailored to address this challenge immediately after the Sprint Review—the Retrospective.

9. Side Gigs

Observation: The Developers worked on issues outside the Sprint Goal, and the Product Owner learned about those for the first time during the Sprint Review.

See Chapter 3, anti-pattern 4, for a detailed description.

10. Showing Off Undone Work

Observation: More often than not, the Developers also show work items that are not "done."

See Chapter 3, anti-pattern 25, for a detailed description.

SPRINT REVIEW ANTI-PATTERNS OF THE STAKEHOLDERS

11. Sprint Review as Approval Gate

Observation: The Sprint Review is used as an approval gate for releasing products to customers.

Background: This Sprint Review anti-pattern is typical for organizations that still cling to a sequential plan-develop-approve-release (non-agile) product delivery cycle. Even some happy agile islands of teams using Scrum correctly are embedded in a traditional project management structure, driven by functional silos, budgeting, and top-down goal setting.

Here are some reasons for stakeholders to cling to a traditional project management approach while employing Scrum Teams:

- **Familiarity and comfort:** Stakeholders may prefer to maintain approval gate practices because they know them. It's human nature to resist change and cling to established methods, especially if they succeed.

- **Illusion of control:** Traditional project management methods often give stakeholders an illusion of control through detailed planning, documentation, and approval gates. They may fear losing this control in an agile environment.
- **Risk aversion:** Some stakeholders may perceive Agile as risky because of its adaptive and iterative nature versus the perceived predictability of using approval gates.
- **Accountability concerns:** In Scrum, the team collectively owns the product. Traditional stakeholders might fear diluted accountability, preferring clear, individual responsibilities.
- **Organizational inertia:** Larger, older organizations often have deep-rooted traditional practices. Changing to Agile may be too disruptive or challenging.
- **Lack of understanding:** Stakeholders may not fully understand Agile and Scrum principles, leading to misconceptions and resistance.

Remedy: It is the prerogative of the Scrum Team to decide which Product Backlog items meet the Definition of Done; no sign-off is required. Furthermore, the Scrum Team decides to release Increment at its discretion. Consequently, the stakeholders do not have a formal say in this decision.

However, pointing to the Scrum Guide and insisting on doing Scrum "right" is usually ineffective in changing this management approach. Instead, try to convince your stakeholders that exercising their tight control over the Scrum Team limits its success, as it delays the closing of feedback loops and thus prolongs the learning process.

Here are some suggestions on how to overcome the anti-pattern in the long term:

- **Demonstrate value and gain trust:** Show the benefits of Scrum in action and build trust by releasing valuable Increments regularly, even under these challenging conditions.
- **Transparency:** Ensure transparency in all aspects of the Scrum process. Make all Scrum artifacts (Product Backlog, Sprint Backlog, Increments) visible to stakeholders, and keep them updated about the progress.
- **Address concerns:** Understand the stakeholders' concerns about moving to Scrum and address them directly. This approach could involve demonstrating how the Scrum Team manages risk and complies with regulatory requirements or how Scrum can improve project predictability.

- **Education:** Provide stakeholders with training or workshops to understand the principles and practices of Scrum. Ensure they know empirical process control's role and how it differs from traditional project management.
- **Coach and support:** Provide ongoing coaching and support to the stakeholders during the transition. This support will help them understand their role in a Scrum environment and how they can contribute to the team's success.
- **Patience and persistence:** Changing deep-rooted practices takes time. Be patient and persistent, reinforcing the benefits of Scrum and addressing concerns as they arise.

12. Internal Stakeholders Do Not Attend

Observation: The Scrum Team runs the Sprint Review for themselves; no internal stakeholders are present.

Background: There may be several reasons why stakeholders do not participate in the Sprint Review; for example:

- They do not see any value in the event.
- They want the information to come to them.
- The Sprint Review conflicts with another important meeting.
- They may not understand the importance of attending the Sprint Review event to the Scrum Team's effectiveness.

In my experience, you need to "sell" the event within the organization. Otherwise, you may lose precious momentum, particularly at the beginning of an agile transformation. By all means, you want to be heard and seen.

Remedy: The most effective way to lure stakeholders to your Sprint Review is the presence of someone from the higher management levels. Rest assured, your stakeholders want facetime with a C-level executive.

Otherwise, if that is not possible, be creative when inviting your stakeholders to the Sprint Review; for example:

- Invite your stakeholders personally to every Sprint Review. Highlight the unique improvements they will get to see and potentially influence.

- Personalize your invitations based on the stakeholder's interest in the product. Make them feel like their specific feedback and presence is crucial for the event.
- Share success stories where stakeholder feedback in previous Sprint Reviews led to significant improvements in the product.
- Write a Sprint report that summarizes the results but also points to important additional information provided during the Sprint Review.
- Use creative marketing techniques; for example, print flyers of the event and leave them in places that your stakeholders frequent.
- Alternatively, place invitations inside of elevators, tea kitchens, or doorways your stakeholders use.
- After the Sprint Review, send a follow-up message thanking them for their participation and summarizing the key points discussed. This will make them feel valued and more likely to attend future reviews.

> ***Practical Tales:*** *While it's not always appropriate, sometimes offering a small token of appreciation can encourage attendance. When I started working as an interim head of product at a prominent startup, Sprint Review attendance was low, so we decided to fix little issues for those stakeholders who attended our Sprint Reviews outside the usual process. It did not take long to convince other stakeholders to join the Sprint Review, too. By that time, we could stop "bribing" them.*

13. No Customers Present

Observation: External stakeholders—also known as customers—do not attend the Sprint Review.

Background: Scrum's empirical process is centered on the idea that the Scrum Team delivers Increments to customers to learn whether the customers consider the Increments valuable by using them and providing feedback. The advantages of their participation are obvious:

- Closing this feedback loop is simpler and faster when customers are participating in the Sprint Review.

- Customers may have ideas of using the Increments that are unknown to the Scrum Team.
- Customers may have knowledge of the problem and solution space that the Scrum Team does not.

Having direct feedback from customers at the Sprint Review inevitably leads to a better outcome and increased value creation.

However, not every internal stakeholder may agree with the Scrum Team on the benefits of customer involvement in Sprint Reviews. Some reasons for this may be, for example:

- **Confidentiality concerns:** Stakeholders may fear that prematurely exposing plans or features to customers might breach confidentiality, especially if there's competitive sensitivity.
- **Control over feedback:** Some stakeholders prefer to control the narrative and filter the feedback that reaches the Scrum Team, hence resisting direct customer involvement.
- **Unfiltered criticism:** Direct customer feedback can sometimes be harsh or demotivating. Stakeholders might worry this could negatively impact team morale.
- **Fear of scope creep:** Stakeholders might worry that customer feedback could lead to scope creep, causing the team to compromise the current Product Goal.
- **Maintaining expectations:** Stakeholders might worry that inviting customers could raise their expectations prematurely, potentially leading to disappointment if future Increments deviate from the outline significantly.
- **Fear of losing authority:** Inviting customers to participate in feedback sessions could shift some power and influence away from internal stakeholders, causing discomfort.

Remedy: Break out of your organization's echo chamber and invite some customers and users to your Sprint Review. Most important, do not let the salespeople object to this idea.

Here are some suggestions on how to address stakeholder concerns regarding the participation of customers in Sprint Reviews:

- **Confidentiality concerns:** You can establish a nondisclosure agreement (NDA) with customers or only invite trusted customers to participate.
- **Control over feedback:** Educate stakeholders about the value of direct, raw feedback in agile product development. Promote transparency and openness as critical factors contributing to a Scrum Team's success.
- **Unfiltered criticism:** Create a safe environment for feedback. Help everyone understand that candid yet constructive criticism is essential for improvement and not a personal attack.
- **Fear of scope creep:** Reiterate the role of the Product Owner in managing the Product Backlog and ordering work based on its contribution to the Product Goal, customer feedback, market conditions, and business needs.
- **Maintaining expectations:** Educate customers that Sprint Reviews present work in progress and that their feedback is invaluable in shaping the final product as the Scrum Team learns how it may look.
- **Fear of losing authority:** Help everyone understand that customer involvement doesn't undermine stakeholders' authority but enriches their and the Scrum Team's understanding of customer needs and market trends.

14. Starting Over Again

Observation: There is no continuity in the attendance of stakeholders.

Background: Long-term relationships develop between the Scrum Team and its stakeholders. It takes time and trust to cultivate the transparency that is essential to sharing valuable feedback. When stakeholders don't attend frequently, these relationships fail to thrive.

Remedy: If this pattern appears, the Scrum Team needs to help stakeholders understand the importance of the Sprint Review and that their attendance is not a mere formality.

To help your stakeholders accept their responsibility toward the Scrum Team, provide stakeholders with training or workshops to help them comprehend the principles and practices of Scrum. Offer ongoing coaching and support to the

stakeholders as well. This support will help stakeholders internalize their role in a Scrum environment and understand how they can contribute to the team's success.

15. Passive Stakeholders

Observation: The stakeholders are passive and unengaged; see Figure 8.3.

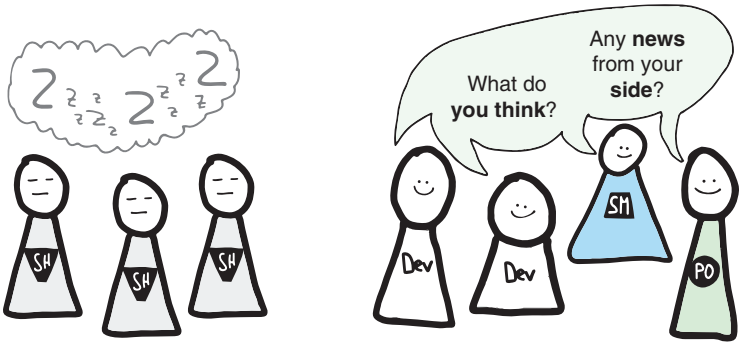

Figure 8.3 There is not much value in a Sprint Review when stakeholders are disengaged.

Background: There are multiple reasons why your stakeholders are less involved than they should be in the Sprint Review; for example:

- **Misalignment of interests:** If the content of the Sprint Review doesn't directly pertain to their interests or responsibilities, they might not feel compelled to participate actively.
- **No visible impact:** If they don't see any action taken based on their previous feedback, they may feel their participation doesn't matter.
- **Intimidation:** In larger groups, some stakeholders might feel intimidated to voice their opinions, especially if dominant personalities are in the room.
- **Lack of trust:** Stakeholders might hesitate to share their honest feedback if there's a lack of trust in the Scrum Team.
- **Lack of preparation:** If the Scrum Team does not brief stakeholders properly on what to expect and how to prepare for the Sprint Review, they might be unable to participate effectively.
- **Communication style:** How the Scrum Team communicates during the Sprint Review might be ineffective for some stakeholders, leading to their disengagement.

- **Prisoners:** Some stakeholders may be forced to participate in the Sprint Review, considering it a waste of their time.

Remedy: Reach out to them, probably via a Retrospective or one-to-one session, to learn more about their lack of engagement. Based on that learning, there are several possibilities the Scrum Team could explore to invigorate stakeholder participation; for example:

- Use interactive formats such as live demos, Q&A sessions, or hands-on testing to encourage participation. For example, let the stakeholders drive the Sprint Review and put them at the helm.
- Organize the Sprint Review as a science fair with several booths where team members introduce solutions to specific problems the Sprint addressed. (Shift & Share[4] is an excellent Liberating Structure microstructure for that purpose.)
- Actively seek stakeholder feedback during the meeting. Ask direct questions to engage quiet participants.
- Break into smaller discussion groups to facilitate more open dialogue. It can feel less intimidating for stakeholders to share their thoughts in a smaller setting.
- Incorporate simple voting tools to gather instant feedback on various topics. This can keep the session interactive and engaging.
- Acknowledge and appreciate stakeholders for their contributions. This recognition can motivate them to participate more actively in future sessions.

Do not expect a simple format change to alleviate the problem; it will take time. (See also the suggestions in anti-pattern 12 of this chapter.)

FOOD FOR THOUGHT

Stakeholders' ignorance: Is it worth having the Sprint Review when internal and external stakeholders are not attending the event but rely on other means of communication, such as reports? Can a Scrum Team convince their stakeholders of the mutual benefits of participating in the Sprint Review? Or could they develop an

4. Henri Lipmanowicz and Keith McCandless, *The Surprising Powers of Liberating Structures: Simple Rules to Unleash the Power of Innovation* (Liberating Structures Press, 2013): 212–16.

asynchronous Sprint Review format that serves the intended purpose outlined in the Sprint Guide?

Forecasting: What effort of the Scrum Team is justifiable to meet the stakeholders' need for accuracy regarding forecasts? Of course, an "accurate forecast" is an oxymoron; all forecasts are inaccurate. However, stakeholders' requests to better understand the possible availability dates of Increments are legitimate, as they need to plan, too. How, then, to best balance those two conflicting positions: revert to deadline-driven development or continue shrugging shoulders while mentioning that the Increment will be ready when it is ready?

Conclusion

Scrum's Sprint Review is a critical Scrum event. It answers whether the Scrum Team is still on track to achieve the Product Goal. Therefore, avoiding the anti-patterns in this chapter can significantly improve a Scrum Team's effectiveness in making customers' and users' lives easier while contributing to a viable organization.

When your stakeholders are less responsive than your Scrum Team would like them to be, gain their attention and make attending the Sprint Review worth their time. Just avoid (Scrum) dogmatism; if it takes a Sprint report sent in advance to stakeholders to get them into the (virtual) room, so be it.

Toolbox

Regarding the following tips and exercises, you will find detailed descriptions in Appendix B:

- **"Agile Metrics—The Good, the Bad, and the Ugly"** points to suitable agile metrics reflecting either a Scrum Team's progress in becoming agile or an organization's progress in becoming a learning organization.
- **"Sprint Review with Distributed Teams"** describes a string of Liberating Structures microstructures that distributed Scrum Teams can use to organize meaningful Sprint Review events.

This page intentionally left blank

9 Sprint Retrospective Anti-Patterns

Introduction

What event could better embody Scrum's principle of continuous improvement than the Sprint Retrospective? I assume all peers agree that even the simplest form of a Retrospective—if held regularly—is far more helpful than having a fancy one once in a while, not to mention having none.

Moreover, I am convinced there is always room for improvement. Hence, learning how to tackle common Sprint Retrospective anti-patterns—see Figure 9.1—that hold back your Scrum Team is a worthwhile investment.

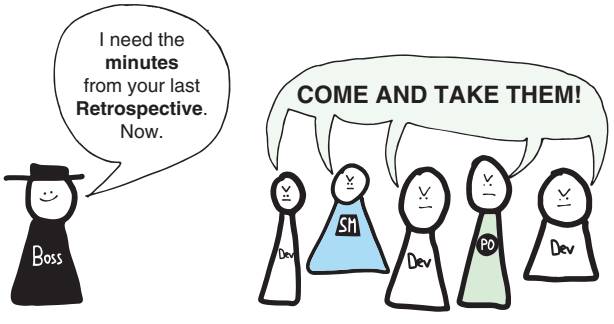

Figure 9.1 Retrospective first principles: What is said during the Retrospective stays in the Scrum Team.

THE SCRUM GUIDE ON THE SPRINT RETROSPECTIVE

According to the Scrum Guide, the Sprint Retrospective serves the following purpose:[1]

- Continuously aims to increase the quality of the Scrum Team's work and effectiveness.
- Regularly inspects interactions, processes, tools, and the Definition of Done for improvement opportunities that the Scrum Team can address promptly.
- Identifies and explores misleading assumptions that hinder the team from creating value.
- Discusses successes, challenges, and problems from the perspective of empiricism.
- Concludes the current Sprint to prepare for the next Sprint.

By any standard, this is a compelling list of the benefits of Retrospectives, addressing the why, the what, the how, and the who.

Suppose a Scrum Team takes inventing on behalf of customers and creating valuable, viable, feasible, and usable products seriously. In that case, it should be highly motivated to diligently close this feedback loop at the team level every Sprint. Furthermore, everyone should look forward to identifying how to make the next Sprint better and more enjoyable by asking, "Are we still progressing toward the team goal?"

Although the Scrum Guide does not specifically talk about the "team goal," each Scrum Team has an implicit goal, their team goal, of improving their ability to achieve Sprint Goals and Product Goals. In this sense, the Sprint Retrospective resembles the Daily Scrum, which is focused on inspecting the progress toward the Sprint Goal, and the Sprint Review, which is focused on inspecting the progress toward the Product Goal.

Unfortunately, many Scrum Teams experience Sprint Retrospectives that do nothing to improve their ability to achieve Sprint Goals and Product Goals. Instead, their Sprint Retrospectives impede their progress and turn a vital practice into a dull, mechanical ritual, devoid of value—no wonder many Developers dread the end of a Sprint.

1. Ken Schwaber and Jeff Sutherland, "Sprint Retrospective," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#sprint-retrospective.

Sprint Retrospective Anti-Patterns

Fortunately, though, there is light at the end of the tunnel, as the Scrum Team members themselves can address many of the Retrospective anti-patterns listed in this section.

Sprint Retrospective Anti-Patterns of the Scrum Team

1. #NoRetro

Observation: There is no Retrospective, as the team believes there is nothing to improve, and the Scrum Master accepts this notion.

Background: There is no such thing as an agile Nirvana where everything is perfect. As people say, becoming agile is a journey, not a destination, and there is always something to improve. Retrospectives are a critical part of Scrum's principle of continuous improvement, which, for example, the Scrum Value of *commitment* reflects. A formalized event like the Retrospective allows the Scrum Team to focus exactly on this challenge.

Remedy: Try to understand why the Scrum Team does not have Retrospectives, for example, by having one-to-ones with team members or going through previous team documents that may point to the reason for the behavior, such as the following:

- Team members have the perception that Retrospectives are not valuable or do not lead to tangible improvements. For example, inadequate facilitation skills may have previously led to unproductive Retrospectives, which caused the team to abandon the practice.
- Despite using Scrum, the organizational culture discourages transparency and open communication, making it difficult for the team to discuss issues openly.
- Team members resist change or are reluctant to engage in open discussions about challenges and areas for improvement.
- Team members fear conflict or negative consequences from discussing problems or suggesting changes.
- Burnout or high workload causes the team to deprioritize Retrospectives in favor of completing work items.

Then ask the team for the benefit of the doubt and facilitate a Retrospective that deserves this designation to convince them of its benefits.

2. Dispensable Buffer

Observation: The Scrum Team cancels Retrospectives if more time is needed to accomplish the Sprint Goal.

See Chapter 3, anti-pattern 26, for a detailed description.

3. The Rushed Retrospective

Observation: The team is in a hurry and allocates much less than the necessary 60 to 120 minutes for a Retrospective.

Background: That is a slippery slope that usually leads to a ritualized ceremony of little value. Don't underestimate a trend you can observe in mediocre Scrum Teams: it starts with rushing Retrospectives, which can lead to skipping them from time to time when a Retrospective would be inconvenient, and ends with not having Retrospectives at all.

Remedy: Don't rush the Sprint Retrospective; a rushed Retrospective is unlikely to unearth any practical improvement issues. The team members will likely consider it a waste of their time, undermining their respect for this critical Scrum event in the future.

Instead, do it right by allocating sufficient time and using effective facilitation techniques to help improve team member engagement.

4. Let's Have the Retro Next Sprint

Observation: The Scrum Team postpones the Retrospective to the next Sprint.

Background: A well-run Retrospective provides a moment of closure that resets everybody's mind so that they can focus on the new Sprint. It provides the Scrum Team with an opportunity to inspect their way of working and identify opportunities to make the next Sprint more effective and fun. This is why Scrum places the Sprint Retrospective before the Sprint Planning event.

The road to self-management is tightly linked with building trust among stakeholders that the Scrum Team will deliver valuable Increments. That building process takes time and perseverance. Also, without the discipline to inspect yourself and act on the results, the team will neither become self-managing nor build trust among stakeholders.

Postponing the Retrospective into the next Sprint doesn't let the team bring the Sprint to a conclusion and may postpone the resolution of one or more critical team issues for the length of another Sprint. Whatever issues are holding the team back will continue to hold the team back.

Remedy: Understand why your teammates want to postpone the Retrospective. Address their concerns, and find out if they consider it unproductive. Then work on improving the structure and conduct of the Retrospective. Using various Retrospective techniques will help keep this crucial event exciting and engaging.

5. Excluding Stakeholders All the Time

Observation: The Scrum Team categorically rejects having Retrospectives with their stakeholders.

Background: The permanent presence of stakeholders or line managers at team Retrospectives is generally not a good idea. However, excluding these two groups from *every* Sprint Retrospective is shortsighted. Doing so deliberately reduces the kinds of feedback the Scrum Team can obtain and may cause them to miss important improvement opportunities.

Many opportunities already exist to provide stakeholders with transparency into what the team is working on. The Sprint Review, of course, is the formal event for stakeholder–Scrum Team communication. In addition, stakeholders can learn by

- Listening in at the Daily Scrum, if the Developers agree.
- Listening in on the Product Backlog refinement.
- Listening in at the Sprint Planning.
- Conversing over Zoom, at water coolers, over coffee, or during breaks.

These events usually focus on what the Scrum Team is doing, not how it is working. There's a good reason for that: how the Scrum Team works is up to the Scrum Team. Nevertheless, stakeholders may have important insights or feedback for the team that can help it improve. Inviting stakeholders to a special Retrospective, occasionally, can provide a forum for sharing these insights.

Remedy: Special Retrospectives that include stakeholders can help to improve collaboration, communication, and trust between the Scrum Team and its stakeholders. I recommend running such stakeholder Retrospectives at least once a quarter.

These special Retrospectives don't take the place of regular Sprint Retrospectives, as a Scrum Team will typically want to discuss things that are either uninteresting or inappropriate to discuss outside the Scrum Team. Exceptions can be made if Scrum Team members unanimously agree.

Toolbox Tip

A Meta-Retrospective is an excellent exercise to foster collaboration within the extended team, create a shared understanding of the big picture, and immediately create valuable action items. It comprises the team members of one or several product teams—or representatives from those teams—and the stakeholders. Participants from the stakeholder side are people from the business as well as customers. (For further information, refer to Appendix B.)

6. Someone Sings

Observation: Someone from the Retrospective participants provides information on the Retrospective to an outsider; see Figure 9.2.

Background: For Retrospectives, the Vegas rule[2] applies: what is said in the room stays in the room. There is no exception to this rule unless all Scrum Team members agree in advance.

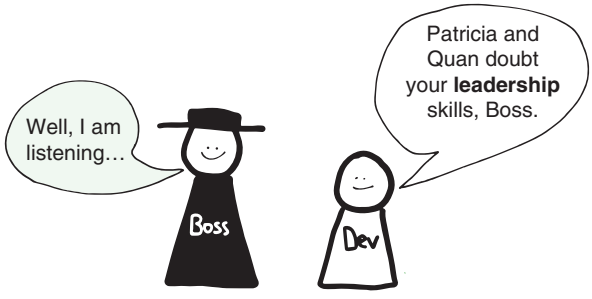

Figure 9.2 There's a snitch on the Scrum Team.

2. *Vegas Rule* is a colloquial term referencing the popular phrase "What happens in Vegas stays in Vegas." It's often used in a meeting or group event context—in this case, our Retrospective—to establish an understanding that none of the participants will share discussions or experiences within the event outside of it. This concept fosters a safe and open environment where participants feel comfortable sharing their thoughts, experiences, and ideas without fear of repercussion or judgment. It encourages trust and open communication, critical elements in team building and conflict resolution. See Urban Dictionary, "Vegas Rule," https://www.urbandictionary.com/define.php?term=Vegas%20Rule.

However, there are—often valid or understandable—reasons why a team member, the "snitch," may leak information from the Scrum Team's Retrospective to outsiders; for example:

- A team member may not trust the confidentiality of the Retrospective process and feel the need to share information with someone outside the team to seek support or validation.
- Someone might seek advice from an external mentor, coach, or more experienced colleague to address a challenge or issue discussed during the Retrospective.
- The team member may not fully understand the importance of maintaining confidentiality within the Retrospective and unintentionally share sensitive information.
- The individual might be frustrated with the team or the process and use the sharing of information to vent their feelings to someone outside the team.
- The team member could be trying to gain leverage or influence within the organization by sharing confidential information with key stakeholders or decision makers.
- The team member might have a personal agenda or ulterior motive for sharing confidential information, such as undermining another team member, furthering their career, or seeking personal recognition.
- Suppose a team member doesn't feel safe to address their concerns due to a perceived lack of psychological safety. In that case, they might disclose confidential information to someone outside the team to seek support or action.
- An individual may believe the team is engaged in unethical or harmful practices and feel compelled to report these issues—whistleblowing.

Remedy: No matter the intention, leaking information to outsiders and disregarding the confidentiality of the team environment is a severe breach of trust and respect, undermining critical ideas that make Scrum work.

Scrum Teams need to react quickly when this happens, not to blame the leaker but to help them understand why this is a critical misstep and to bring them back into the team. Doing so may also require the Scrum Team to resolve the issues that led the person to leak.

7. UNSMART Action Items

Observation: The team chooses to tackle UNSMART actions, or acts of improvements that have no chance of materializing or are of lesser value than possible other actions.

Background: George T. Doran created the SMART acronym for reasonable goals:[3]

S—Specific
M—Measurable
A—Achievable
R—Relevant
T—Timeboxed

Scrum Teams picking SMART action items communicate their determination to continuously improve by focusing on the most important tasks—an essential capability for self-managing teams. However, if the team picks UNSMART action items, it likely sets itself up for failure and may thus contribute to a bias that "Agile" is not working in the organization.

Remedy: If your Scrum Team struggles with applying the SMART framework to their action items, consider running a workshop that helps map the framework to your team's situation, for example, their degree of autonomy or organizational constraints. Becoming proficient at creating SMART action items is a process and takes time to inspect and adapt. Once you have cleared that hurdle, your Scrum Team will find the concept most beneficial to improve its way of working. Learn more from "INVEST in Good Stories and SMART Tasks."[4]

8. #NoAccountability

Observation: The team members accepted action items at the last Retrospective. However, no one took responsibility for the delivery. As a result, the team revisits several issues at its next Retrospective, losing valuable time.

---

3. See George T. Doran, "There's a S.M.A.R.T. Way to Write Management's Goals and Objectives," *Management Review* 70, no. 11 (November 1981): 35–36.
4. Bill Wake, "INVEST in Good Stories, and SMART Tasks," XP123, August 17, 2003, https://xp123.com/articles/invest-in-good-stories-and-smart-tasks.

Background: If the "team" is supposed to fix X, everyone will probably rely on their teammates to handle it. This form of hoping for the best is most likely to occur when taking care of the action items means tedious, unglamorous work that won't get much attention.

Remedy: For the area of these tasks, there is a simple approach to solving the problem: make sure someone on the team makes themselves responsible for ensuring the action items get done. They might enlist the help of others, but they are responsible.

This concept of a "directly responsible individual"[5] or DRI, is borrowed from Apple.

9. What Improvement?

Observation: The team does not check the status of the action items from previous Retrospectives.

Background: The sibling of autonomy is accountability. If you are not following up on what you wanted to improve, why care about picking action items in the first place?

Understanding how the team is dealing with self-chosen action items represents valuable information to Scrum Masters regarding the respective team's capability to self-manage. It also points to areas where more teaching and coaching would be helpful.

Remedy: A proven tactic to foster accountability among team members regarding action items is to revisit the tasks from previous Retrospectives at the beginning of subsequent Retrospectives. Think of it as an embedded Daily Scrum-like 5-minute event to better understand where the team is heading on its path to continuous improvement.

Mature Scrum Teams do this automatically; a junior team may require some gentle guidance from the Scrum Master, politely inquiring whether they can support the closing of any open action item.

---

5. Apple's directly responsible individual concept refers to the person who is ultimately accountable for a specific task, project, or decision within the company. The DRI drives the project forward, ensures meeting deadlines, and coordinates with other team members or stakeholders as needed. This approach aims to encourage a sense of ownership and responsibility among employees and minimize confusion regarding who is responsible for what. Recommended reading: Adam Lashinsky, *Inside Apple: How America's Most Admired—and Secretive—Company Really Works* (Business Plus, 2012).

10. #NoDocumentation

Observation: No one is taking minutes for later use.

Background: A Retrospective is a substantial investment and should be taken seriously. Documentation provides multiple benefits:

- A written record of action items and who is responsible can increase responsibility and commitment, ensuring that the team realizes improvements. (See also the "DRI" in the #NoAccountability anti-pattern in this chapter.)
- Notes help document action items identified during the Retrospective, making it easier for team members to remember and follow up on them (see preceding anti-pattern 9, What Improvement?).
- Retrospective notes can serve as a reference for future Retrospectives, allowing the team to identify patterns, review past discussions, track progress, and learn from their experiences.
- Notes can be shared with other Scrum Teams or stakeholders within the organization, fostering a culture of continuous learning and improvement. They are essential building blocks of a community of practice.
- Retrospective notes can be a valuable resource for new team members, providing them with insights into the team's history, challenges, and areas of improvement.

Remedy: At the beginning of a Sprint Retrospective, a Scrum Team shall identify someone who will take notes on behalf of the team for later use. I recommend rotating the role regularly so that everyone on the team shares the task of recording minutes.

11. Extensive Complaining

Observation: The team members use the Retrospective primarily to complain about a situation and assume the victim's role without taking action to change.

Background: Change requires reflection, and used sparingly, venting has value. Once you have identified critical issues, however, shift your focus to doing something about them.

Several reasons for this attitude come to mind:

- Team members may feel they have little control over their work environment and lack the ability to effect change.

- Team members may lack a sense of ownership for their work or the team's success.
- Team members may lack the skills or experience necessary to effectively identify and address issues.
- Team members may have unresolved conflicts with other team members that make it difficult to collaborate to find solutions.

If assuming the victim role is limited to one or two team members, it is most likely a personal issue. If all the team members act this way, it typically is a much more severe situation. This form of complaining often happens in organizations that formally adopted Scrum but do not practice it.

Remedy: When team members play the role of victim, they make themselves unable to resolve the source of conflict. Nevertheless, merely dismissing the issue as misguided behavior of professional people won't solve the problem.

In the case of individuals, I suggest reaching out to them to understand the underlying motives for their way of communication and coaching them on how to overcome their issues.

If the whole team is assuming this behavior, a simple individual or group coaching won't help. (Neither will joining your teammates and becoming sarcastic along with them.) As an organizational matter, consider reaching out to the leadership level and asking for their support to help the team turn around and accept their responsibilities.

Sprint Retrospective Anti-Patterns of the Scrum Master

12. Groundhog Day

Observation: The Sprint Retrospective never changes in composition, venue, or length.

See Chapter 1, anti-pattern 14, for a detailed description.

13. Prisoners

Observation: Some team members participate only because they are forced to join.

Background: If team members are afraid of negative repercussions in the case of their absence and hence decide grudgingly to show up, it defies any Retrospective's first principle: all participants need to attend because they genuinely care for their team's continuous improvement and take their commitment seriously. That motivation needs to be intrinsic. Consequently, no team member can force a teammate to participate in a Retrospective.

Some reasons why Scrum Team members may feel like prisoners are, for example:

- **Lack of psychological safety:** Team members may feel they can express their thoughts, concerns, or ideas only with fear of judgment, ridicule, or punishment, making them feel trapped and anxious during meetings.
- **Pressure to conform:** Some team members may feel others expect them to go along with the majority opinion, even if they disagree. This pressure to conform can create a sense of imprisonment.
- **No clear purpose:** Team members don't see the Retrospective's value, as nothing changes no matter the outcome. Nevertheless, they may feel obliged to attend without seeing the benefit, leading to resentment.
- **Lack of respect for personal time:** Team members can feel trapped and obligated if Retrospectives are scheduled at inconvenient times or intrude on personal time.
- **Unresolved conflicts:** Interpersonal or team conflicts that the Scrum Master fails to address can create an uncomfortable environment where team members feel they have to walk on eggshells.

Remedy: Don't pressure anyone to take part in a Retrospective. Instead, make it worth their time. The drive to continuously improve as a team needs to be fueled by intrinsic motivation, not fear or force.

Some first steps include the following:

- **Foster psychological safety:** Make the Retrospective a safe place where everyone can share their thoughts, feelings, and ideas without fear of judgment or ridicule. Encourage open dialogue and respect for all perspectives.
- **Promote autonomy:** Rather than forcing attendance, encourage it by creating a Retrospective that team members find valuable, engaging, and rewarding. Promote a culture of collective ownership and responsibility for team improvement.

- **Clarify purpose and value:** Ensure every team member understands the goal of the Retrospective and its value for continuous improvement. Make it clear that it's not about blaming or criticizing but about learning and adapting.
- **Respect personal time:** Schedule the Retrospective at a time that works for everyone, and respect the timebox to ensure the meeting doesn't drag on unnecessarily.
- **Address conflicts:** If interpersonal issues or conflicts hinder participation, work to address them respectfully and constructively. Do so outside the Retrospective to separate the conflict from the event.
- **Involve the team in meeting the design:** Give team members a chance to contribute to the Retrospective design, for example, by rotating the facilitator role or letting the team choose the format or activities.
- **Focus on actionable outcomes:** Make sure every Retrospective results in clear, actionable improvements, providing the team members with tangible evidence of the value of their participation.

Toolbox Tip

If you are unsure whether some of your team members participate only involuntarily, Esther Derby and Diana Larsen's ESVP activity exercise is an excellent way to understand the situation.[6]

14. The Blame Game

Observation: The Retrospective is an endless cycle of blame and finger pointing; see Figure 9.3.

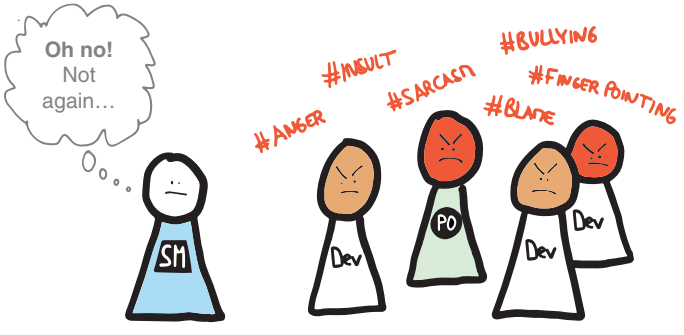

Figure 9.3 The team wins together; the team fails together. So, playing the blame game is a lose-lose situation for everyone.

Background: Dispensing blame does not solve any problems. On the contrary, blaming others will likely escalate a misunderstanding or unfortunate incident into a full-blown conflict. Ultimately, the conflict will lead to faction-building and eliminate trust among the team members. Furthermore, the original problem will be quickly forgotten as every faction focuses on being right and making the other one lose.

The team wins together; the team fails together. The blame game documents both the failure of the Scrum Master as the likely facilitator of the Retrospective and the Scrum Team's lack of maturity and communication skills and how much they despise Scrum Values. Consequently, the Scrum Team will degrade into a group.

Remedy: Building trust may take weeks or months; destroying it takes barely more than seconds.

There are some things you can try, to rebuild trust among your teammates:

- Advocate Scrum Values and encourage open and honest communication, emphasizing the importance of focusing on issues and solutions rather than assigning blame. Retrospectives are a safe space for open discussion without fear of repercussion.
- Share your failures openly and embrace a personal level of vulnerability to help the team live up to Scrum Values.
- Create an environment where team members feel comfortable sharing their thoughts, concerns, and mistakes. Encourage your teammates to speak up and provide support when needed.
- Guide the Scrum Team in identifying problems and potential improvements rather than personal criticism. Use techniques like the Five Whys or some other root-cause analysis to explore issues more deeply and find underlying causes.
- Help team members understand their role in contributing to the team's success and addressing challenges. Help them live up to Scrum Values, take responsibility for their actions, and commit to improving.
- Engage the team in exercises or games that foster collaboration, communication, and trust-building.

6. Esther Derby and Diane Larsen, *Agile Retrospectives: Making Good Teams Great* (Pragmatic Bookshelf, 2006): 44–46. See also "ESVP," Retromat, https://retromat.org/en/?id=1.

- Support the Scrum Team members to provide constructive feedback to each other, both during and outside of the Retrospective. This openness can help prevent misunderstandings and promote a culture of continuous improvement.
- Ensure the Scrum Team follows through on the agreed-upon action items and tracks progress. This practice demonstrates that the team takes concerns and suggestions seriously and can help rebuild trust.
- Recognize and celebrate individual and team successes. There is always a reason to have a team breakfast or lunch.

If you can, avoid your Scrum Team's descent into the blame game at all costs. No matter the effort to regain trust, it may likely never be the same again.

15. No Psychological Safety

Observation: One or two team members dominate the Retrospective.

See Chapter 1, anti-pattern 19, for a detailed description.

SPRINT RETROSPECTIVE ANTI-PATTERNS OF THE ORGANIZATION

16. No Suitable Venue

Observation: There is no good place available to run the Retrospective.

Background: The least appropriate place to have a Retrospective is a meeting room with a rectangular table surrounded by chairs. And yet it is the most common venue to have a Retrospective.

Instead, a suitable space for conducting a Retrospective should be spacious, comfortable, well lit, and well ventilated, with amenities such as a whiteboard, flipcharts, markers, and sticky notes. Moreover, it should offer privacy and be free from interruptions.

Seating arrangements can be made to suit the planned Retrospective format best and should promote equality and active participation, avoiding hierarchical setups. Alternatively, you can eliminate any seating arrangement. The location must be accessible to everyone, and the room should be available for the entire duration of the Retrospective to avoid rushed sessions.

Remedy: Running a retrospective requires space. If this space is not available, you should become creative and go somewhere else:

- Use another team's area.
- Occupy the organization's cafeteria or café.
- If the weather is fine, going outside can help a team to relax and open themselves up to new perspectives.
- Rent a suitable space somewhere else.
- If that is not working, for example, due to budget issues, remove at least the table so you can sit/stand in a circle.

Be creative. Learn more: "Agile Workspace: The Undervalued Success Factor."[7]

17. Line Managers Attend

Observation: Line managers regularly participate actively in Scrum Team–internal Retrospectives, raising issues and pointing to possible solutions; see Figure 9.4.

Figure 9.4 There is no psychological safety when line managers participate in all team Retrospectives (which does not rule out their occasional participation upon invitation by the Scrum Team).

Background: This is among the worst Sprint Retrospective anti-patterns I can imagine. It turns the Retrospective into an unsafe place. And who can expect that an unsafe place would trigger an open discussion among the team members?

Nevertheless, and despite collision with basic Scrum principles, your line managers may consider their participation in Retrospectives to be important and necessary; for example:

- The manager may genuinely believe that their participation in the Retrospective will provide valuable insights, guidance, or support to the team, helping them improve and grow.
- The manager might be aware of conflicts or issues within the team and may want to intervene directly to help resolve these problems and support the team's growth.
- The manager may see Retrospectives as an opportunity to gather valuable insights into the team's performance and challenges, using this information to make informed decisions at the management level for the team's benefit.

Of course, they may also harbor less noble intentions:

- The manager may not trust the team to identify and address issues independently, believing their presence is necessary to ensure accountability and follow-through.
- The manager has a perceived need to maintain control over the team's processes, decisions, and direction, believing their involvement is critical to the team's success.
- The manager may feel threatened by the team's autonomy and the prospect of change, leading them to assert their presence to preserve the status quo.
- The manager may not fully understand the principles of Scrum and the importance of self-management; they may believe that the Scrum Team requires their input for a productive Retrospective.

Remedy: In these situations, to help the team learn to self-manage, I recommend that Scrum Masters work with the respective line managers to find alternative ways to support the team and share information. It is essential for line managers to respect the team's autonomy and help foster a safe environment for continuous improvement.

7. Stefan Wolpers, "Agile Workspace: The Undervalued Success Factor," November 13, 2016, https://age-of-product.com/agile-workspace.

18. Reporting Structures within the Scrum Team

Observation: A junior Developer reports to a senior Developer on the same Scrum team.

Background: There are no hierarchies within Scrum Teams. No one has the authority to tell anyone else what to do, when, or how—for a good reason: in a complex environment, there are no experts. This egalitarian approach serves to improve finding solutions, foster collaboration, and help to build trust among team members.

Introducing a reporting hierarchy reduces psychological safety, as the junior Developer is less likely to share openly with the rest of the Scrum Team, including their superior, their concerns, suggestions, or observations, to avoid a career-limiting step.

Remedy: Scrum Teams thrive on openness, respect, and courage; the Retrospective is the place to exercise all of them. Consequently, this situation requires the immediate attention of the Scrum Master as it instantly turns the Retrospective into an awkward and significantly less helpful event as the subordinate is unlikely to share openly what might be affecting their superior or their career.

Ultimately, to avoid future disasters of this kind, reach out and help the line management and human talent department to understand the problem they are causing by creating reporting hierarchies within Scrum Teams: an organization that utilizes Scrum to become agile should avoid this problem proactively.

> ***Practical Tales:*** *I was working with a Scrum Team at a prominent, multinational utility company where we faced this problem. Consequently, we ran two Retrospectives: one with the complete team, including a restrained junior Developer, and a one-to-one between the junior Developer and me, where he shared what was going on from his perspective. Despite addressing the issue regularly at different levels, the problem was only solved when the senior Developer changed jobs.*

FOOD FOR THOUGHT

Can a Scrum Team decide to abandon Scrum? What if a Scrum Team decides during the Retrospective to quit Scrum (see Figure 9.5)? They are self-managing, aren't they?

According to a July 26, 2021, LinkedIn poll, 79 percent of 470 participating professionals supported the notion that a Scrum Team can decide to abandon Scrum.[8]

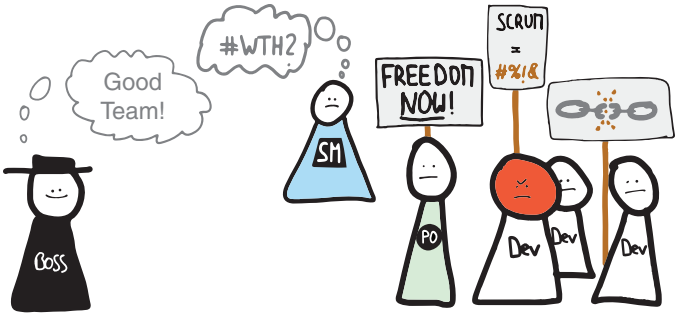

Figure 9.5 Does the Scrum Guide empower *self-managing* Scrum Teams to decide on quitting Scrum?

Can Retrospectives become obsolete? Imagine an outstanding Scrum Team. Is it possible to progress to such a level that a formal team event is no longer required? What if the Scrum Team performs a format of continuous Retrospective, addressing all issues on the spot or ignoring them deliberately in an outspoken self-managing fashion?

Conclusion

There are many ways in which a Sprint Retrospective can be a failure, even if it looks suitable at first glance. Following are my top three Sprint Retrospective anti-patterns:

- Not making the Retrospective a safe place
- Forcing team members to participate in an exercise they consider a waste of their time
- Not taking the Retrospective seriously in the first place

If you embrace the Scrum Team's ability to self-manage and continuously improve, investing in helping them to utilize Sprint Retrospectives and helping them to avoid Sprint Retrospective anti-patterns is a worthwhile investment.

8. Stefan Wolpers, "Can a Scrum Team decide to ditch Scrum?" LinkedIn Poll, July 26, 2021, https://www.linkedin.com/posts/stefanwolpers_let-us-delve-into-a-controversial-topic-activity-6825357507611934720-ZP9z.

Toolbox

Regarding the following tips and exercises, you will find detailed descriptions in Appendix B:

- **"Meta-Retrospectives—How to Get Customers and Stakeholders Onboard"** details an excellent Retrospective format to foster collaboration within the extended team, including your team's stakeholders, to build a shared understanding of the big picture and create valuable action items.
- **"Retrospectives with Distributed Teams"** describes a string of Liberating Structures microstructures that distributed Scrum Teams can use to organize meaningful Retrospectives.

10 Product Backlog and Refinement Anti-Patterns

Introduction

Scrum is a helpful tactical framework to build products, provided you identify what is worth making in advance. But even after a successful product discovery phase, you may struggle to create the right thing if your Product Backlog is not up to the job. Sadly, Scrum is equally well suited to collaboratively build something no one wants or needs in small steps: garbage in, garbage out.

The Product Goal and, subsequently, the Product Backlog are critical to prevent this from happening. As I cover the Product Goal in a separate chapter, the following text points to common Product Backlog anti-patterns—including the refinement process—limiting your Scrum Team's success.

THE PRODUCT BACKLOG ACCORDING TO THE SCRUM GUIDE

First of all, let's have a look at the current edition of the Scrum Guide on the purpose of the Product Backlog. The main characteristics of the Product Backlog are as follows:[1]

- The Product Backlog is a dynamic, ordered list of work the Scrum Team considers necessary to accomplish the Product Goal, serving as the Scrum Team's single source of work.
- Refinement is an ongoing process of breaking down and defining Product Backlog items into smaller tasks, adding details like description, order, and size.
- Developers are responsible for sizing; they may receive guidance from the Product Owner on understanding and selecting trade-offs.

Scrum, being a tactical framework, is good at effectively turning validated hypotheses into Increments and bringing those to customers to close the learning loop. Coming full circle this way allows for the inspection of the initial assumption and decision process to build an Increment. Subsequently, the Scrum Team adapts the Product Backlog based on the feedback to prepare for the next Sprint.

However, Scrum is not good at formulating hypotheses and running experiments, validating, or falsifying them, otherwise referred to as *product discovery*. That is not tactics but part of operations, positioning the Scrum Team for success. If you look at the flow diagram in Figure 10.1, you see that Scrum's part—tactics—is on the right side, while the product discovery part—operations—is on the left. There are ample opportunities to deal with the left part of the process, such as lean startup, design thinking, design sprint, Lean UX, and dual-track Agile,[2] to name a few.

At any given time, a Scrum Team needs to practice product discovery and product delivery (or product development) simultaneously and continuously. However, this necessity does not mandate squeezing all work items into the Product Backlog. On the contrary, teams perform below their capabilities when they do not adhere to maintaining two different artifacts for both parts of the process.

1. Ken Schwaber and Jeff Sutherland, "Product Backlog," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#product-backlog.
2. Jeff Patton, "Dual Track Development Is Not Duel Track," May 10, 2017, https://www.jpattonassociates.com/dual-track-development.

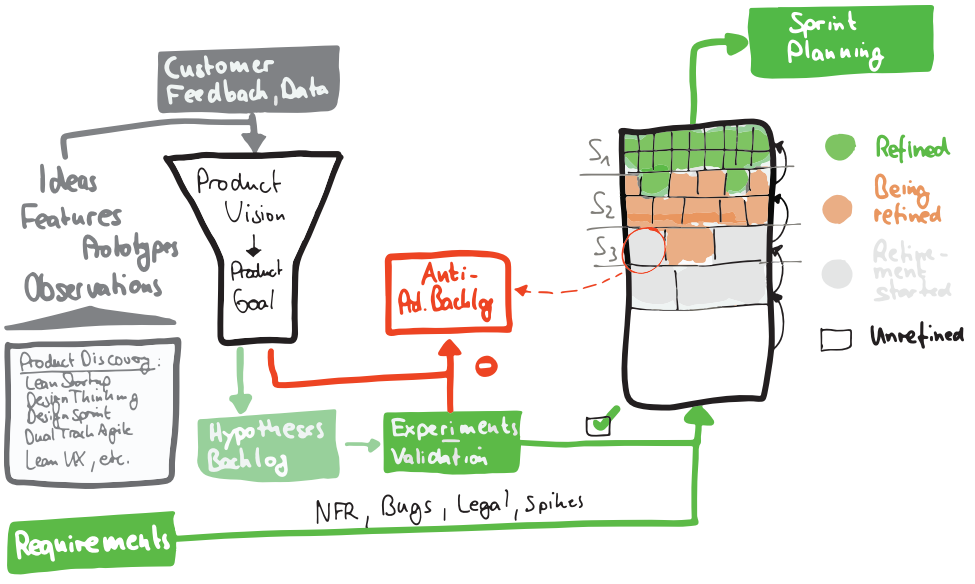

Figure 10.1 Garbage in, garbage out: If you fail to build a system to feed valuable work items into the Product Backlog, Scrum will be equally effective in building stuff no one needs.

The first is a transparent artifact to keep track of ideas, hypotheses, experiments, and outcomes, comprising two repositories: the hypotheses backlog and the anti-product backlog. While the hypotheses backlog holds all the ideas that support accomplishing the Product Goal with a high probability, the anti-product backlog contains falsified hypotheses and other items that a Scrum Team decides not to pursue at the moment. Both repositories are helpful augmentations to Scrum and support the Scrum Team to focus on accomplishing the Product Goal.

The second artifact is, of course, the Product Backlog. An effective Scrum Team regularly invests in keeping this artifact *actionable*. An actionable Product Backlog reflects the continuous effort of the Scrum Team in product discovery in general—developing an informed opinion of which idea may be valuable to the customers—and refining those validated hypotheses into proper Product Backlog items, ready to be taken on in a Sprint. Also, an actionable Product Backlog allows the Scrum Team to run an effective Sprint Planning at a moment's notice.

At this point, everyone on the Scrum Team

- Understands why the team pursues this opportunity over others,
- Knows what the Developers shall build,
- Usually knows how they shall accomplish the work, and
- Probably knows who will work on it.

Typically, and depending on your industry, product, or service, the size of an actionable Product Backlog comprises three to six Sprints of work. Given that the Product Owner orders Product Backlog items by value, the Scrum Team also understands when they might work on any of the individual Product Backlog items.

Always, when we talk about the Product Backlog, we need to add the phrase "based on what we know now," at least in your thoughts, as the Product Backlog is emergent, and any new learning may change its composition and order.

COMMON PRODUCT BACKLOG ANTI-PATTERNS

Despite being relatively straightforward, the process of creating and refining a Product Backlog often suffers from various anti-patterns. Representing great opportunities to support your Scrum Team in creating value when solved, I have identified multiple categories of Product Backlog anti-patterns.

GENERAL PRODUCT BACKLOG ANTI-PATTERNS

1. Prioritization by Proxy

Observation: A single stakeholder or a committee of stakeholders decides on the nature of the Product Backlog. The Product Owner liaises for this or these individuals, passing their ideas to the rest of the team.

Background: In theory, Scrum's strength builds on the Product Owner's elevated position: Product Owners formulate and communicate Product Goals, which inform the composition and the ordering of any Product Backlog.

Consequently, the Product Owner is the sole person to decide what tasks become Product Backlog items and probably turn into Increments, or, as the Scrum Guide puts it, "The Product Owner is one person, not a committee."[3]

3. Ken Schwaber and Jeff Sutherland, "Product Owner," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#product-owner.

Take away that empowerment, and Scrum quickly turns into a waterfall 2.0 process[4] with iterations and feedback loops; see Figure 10.2.

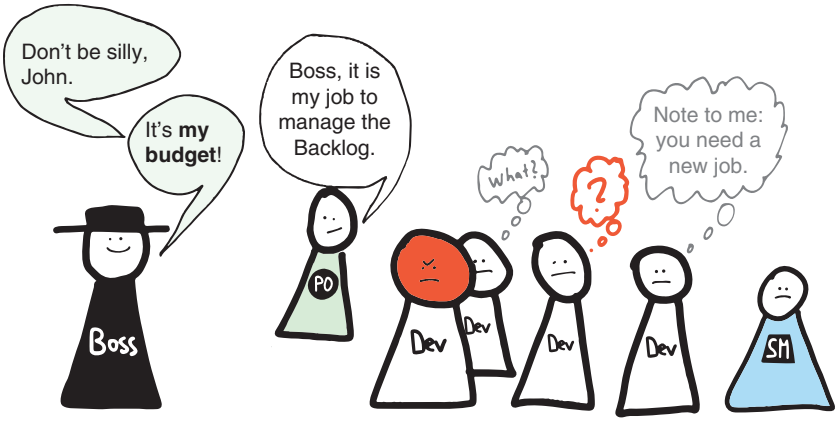

Figure 10.2 Remove the Product Owner's prerogative to manage the Product Backlog, and Scrum becomes a powerful waterfall 2.0-like process.

Remedy: We are not paid to practice Scrum but to solve our customers' problems within the given constraints while contributing to the organization's sustainability. Maybe your organization will identify its way of working by appropriating some of Scrum's principles and techniques—which is fine.

However, if you intend to reap the benefits of Scrum—delivering timely valuable outcomes to customers with less waste—respecting the Product Owner role in its entirety is crucial. As is often the case with a significant change, you cannot initiate this transition from a traditional, stakeholder-driven approach to Scrum by decreeing it. On the contrary, you need to include everyone, give everyone a voice, and convince them that an empowered Product Owner is beneficial to their interests, too. Scrum will not live up to its potential when stakeholders treat the team like an internal development agency and the Product Owner as the one turning stakeholders' requirement documents into Product Backlog items.

4. The waterfall model is a sequential project management approach traditionally used in software development. Projects are divided into distinct, linear stages, each dependent on the one before it. Progress cascades like a waterfall from one phase to the following—requirements, design, implementation, verification, and maintenance—without revisiting previous stages. However, it is less flexible to change mid-project compared, for example, to Scrum. For a deep dive, consider reading Winston W. Royce, "Managing the Development of Large Software Systems," in *Technical Papers of Western Electronic Show and Convention (WesCon)*, August 25–28, 1970, Los Angeles, USA.

2. Oversized Product Backlog

Observation: The team creates a massively comprehensive Product Backlog; see Figure 10.3.

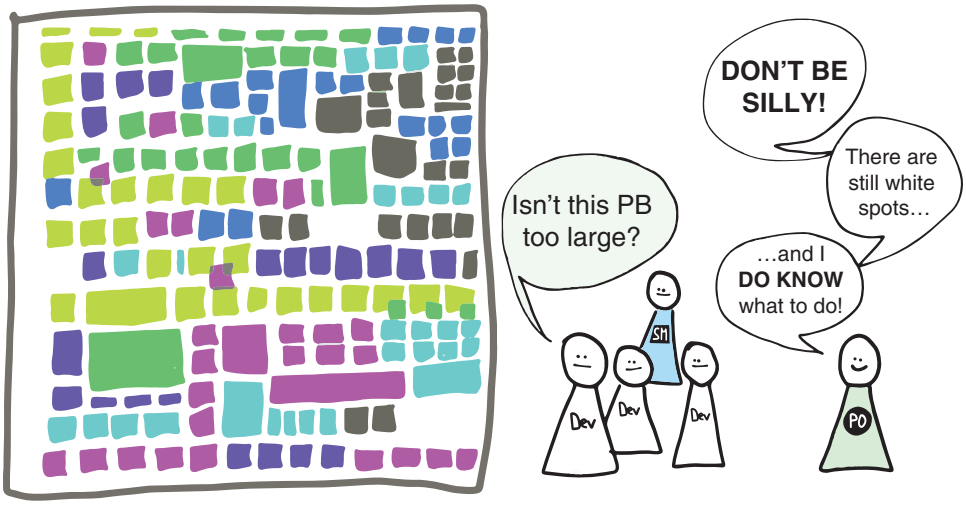

Figure 10.3 Oversized Product Backlogs aren't just waste—they can hide a sunk cost fallacy trap.

Background: Building a comprehensive list of all work items for a release is a waste of the team's time. In a complex environment with a lot of uncertainty about customer needs or the best way to satisfy them, detailing all the work that the team needs to do simply creates waste in the form of Product Backlog items that are refined but will never be released, as more valuable items are later identified. Moreover, investing in refining Product Backlog items that may turn out not to be useful may lead you into the sunk cost fallacy trap,[5] causing you to hang on to parts of the solution that feedback tells you aren't needed, and preventing you from moving on to better solutions.

5. The sunk cost fallacy is a cognitive bias that causes individuals to continue to invest in a project or decision based on prior investments rather than assessing its current and future value. This tenacity can lead to irrational decision-making, as people may persist with unfavorable choices due to their previous investments. Source: H. R. Arkes and C. Blumer, "The Psychology of Sunk Cost," *Organizational Behavior and Human Decision Processes* 35, no. 1 (1985): 124–40.

Remedy: The Product Backlog should be concise and actionable, reflecting the necessary work for the current Product Goal without investing too much effort into refinement in advance.

Depending on the kind of Product Goal, a helpful rule of thumb is to limit the Product Backlog to items sufficient for three to six Sprints. While this work is under way, the Scrum Team should limit their refinement work on Product Backlog items to no more than the next two to three Sprints. This helps to keep everyone aligned with the Product Goal and Sprint Goals by limiting the time they spend detailing Product Backlog items that may change anyway as the Scrum Team learns more about customer needs and the best ways to meet them.

3. 100% in Advance

Observation: The organization tasks the Scrum Team with re-creating a legacy application one-for-one on a new technology base. So, the Scrum Team prepares the complete Product Backlog before they head for the first Sprint. After all, it is the same set of functionality.

Background: This approach is a classic rookie mistake, as it assumes that the legacy system already perfectly meets the needs of its customers or users; it rarely does. Even if it perfectly met their needs when it was created, their needs have probably changed since then. The world certainly has.

Remedy: Rebuilding a legacy application offers an outstanding opportunity not only to replace the technical foundation but also to

- Analyze and improve existing processes,
- Improve usability, and
- Make the application simpler to use while unearthing hidden value.

Never let such a splendid opportunity to improve value creation go to waste just because someone believes that your Scrum Team simply needs to rebuild the old application.

Practical Tales: *I worked with a Scrum Team for a large utility. The organization tasked the team to rebuild a mandatory application by insourcing it from a service provider. (The UK subsidiary had to offer the application to its customers for legal reasons.) There also was a legitimate deadline due to contractual requirements. To make a long story short:*

- *We delivered two weeks early.*
- *We spent less than 85 percent of the allocated budget.*
- *We streamlined processes and made them self-explanatory.*
- *We improved the overall usability.*

We were the first team to accomplish such a task in such a manner. We received a lot of praise from the organization, and the team kept working to further improve the application, which moved from liability to competitive advantage in the eyes of the management.

4. Outdated Items

Observation: The Product Backlog contains items that haven't been touched for months.

Background: The Product Backlog reflects the most valuable work for the Scrum Team to be working on at any given moment. In a complex environment, what is most valuable changes regularly, and that change is reflected in the composition and ordering of Product Backlog items. Just because the Product Owner deemed a Product Backlog item valuable at a particular moment in the past does not mean that the item is still valuable today.

If the Product Owner hoards Product Backlog items, never removing them when they are no longer valuable, the Product Backlog will not reflect the highest items of value. Those outdated Product Backlog items turn from signal to noise, possibly obscuring the actual value of other items.

Remedy: Product Backlog items that are no longer valuable should be removed from the Product Backlog. The previously mentioned anti-product backlog is a suitable place to store these items for later consideration.

5. Everything Is Detailed and Estimated

Observation: All Product Backlog items are thoroughly detailed and estimated.

Background: That is too much upfront work and bears the risk of misallocating the team's time. Product Backlog items need to be refined only to the point where the Scrum Team feels comfortable turning these items into Increments. The extent to which the team refines will depend on their skills and shared understanding of the Product Backlog items being refined.

As noted earlier, the Scrum Team also needs to refine only the Product Backlog items that they think they will work on in the next one to three Sprints. All of those items are at different stages of refinement:

- Those items considered to be candidates for the upcoming Sprint are well-understood.
- The work on the items for the next but one Sprint are in the early stages.

Remedy: While refining feels like real work, it's only useful to the extent that the refine Product Backlog items get built. If things change and they never get built into an Increment, the work spent refining them is simply waste. As a result, don't refine too far ahead.

6. INVEST?

Observation: The Scrum Team does not apply the INVEST principle[6] to Product Backlog items.

Background: Failing to create independent, valuable, and small work items increases the risk of implementation failure during the Sprint. Consequently, a Scrum Team is well advised to invest—no pun intended—in a proper Product Backlog refinement practice to address possible problems before they start working on Product Backlog items during the Sprint.

Remedy: Bill Wake's INVEST principles—Independent, Negotiable, Valuable, Estimable, Small, and Testable—have proven helpful in guiding Scrum Teams through the process of Product Backlog item refinement. Although the INVEST

6. Bill Wake, "INVEST in Good Stories, and SMART Tasks," August 17, 2003, https://xp123.com/articles/invest-in-good-stories-and-smart-tasks.

mnemonic was created for software development, Scrum Teams can quickly adapt and apply the principles to other products.

7. Missing Acceptance Criteria

Observation: There are Product Backlog items that need additional acceptance criteria.

Background: All Product Backlog items must meet the Definition of Done and—in some cases—additional specific requirements to be considered done: the acceptance criteria. Acceptance criteria defines conditions that a Product Backlog item must meet so we can regard it as complete. They promote alignment between Developers and the Product Owner about what the Product Backlog item must achieve.

Having acceptance criteria for Product Backlog items when they are being refined is unnecessary, but it makes refinement easier. By the end of refinement, the Product Owner and the Developers need to agree on acceptance criteria for all of the refined Product Backlog items.

Remedy: If the Definition of Done does not cover all requirements for a specific Product Backlog item, acceptance criteria for the Product Backlog item must be defined during the refinement process. Including a clause for that purpose in the Definition of Done is helpful, such as "The Product Backlog item meets all previously agreed-upon acceptance criteria."

8. Too Detailed Acceptance Criteria

Observation: Product Backlog items have an extensive list of acceptance criteria.

Background: This is the other extreme compared to the previous Product Backlog anti-pattern. The Product Owner covers every conceivable edge case or eventuality without prior negotiation with the Developers. The problem has several aspects:

- It is rarely possible to cover all edge cases.
- However, we only understand which are relevant with feedback from real users.
- Then we need to consider opportunity costs and return on investment: Does it make sense to address all the edge cases?

Finally, with an extensive list of acceptance criteria, the Product Owner may move from their side of the negotiation—the why and what—to the Developers' side, the

how. There is a risk that Product Owners start telling the Developers how to implement a Product Backlog item, which is the Developers' prerogative.

Remedy: Typically, three to five acceptance criteria are more than sufficient. If you believe that you need more criteria, this may indicate that the work item is still too large and needs splitting. Talking to your Developers will clear the situation.

PRODUCT BACKLOG ANTI-PATTERNS OF THE PRODUCT OWNER

9. No Research

Observation: The Product Backlog contains no, or only a few, timeboxed research tasks, or spikes.[7]

See Chapter 2, anti-pattern 13, for a detailed description.

10. Last-Minute Product Backlog

Observation: The Product Owner and the rest of the Scrum Team do not invest adequately in Product Backlog creation and refinement. Consequently, they rush to fill the Product Backlog immediately before the Sprint Planning to ensure there is enough work for the upcoming Sprint to meet stakeholder and management expectations.

See Chapter 2, anti-pattern 14, for a detailed description.

11. Storage for Ideas

Observation: The Product Owner uses the Product Backlog as a repository of ideas and requirements.

Background: A large Product Backlog, for some, is a sign of a "good" Product Owner:

- The Product Owner is fully transparent.
- Stakeholders feel respected that their requirements made it into the Product Backlog.
- It proves the Scrum Team's usefulness to the organization.

7. Some people also refer to them as "spikes," a term from Extreme Programming. A spike is a timeboxed exploration or investigation to reduce uncertainty or risk surrounding a technical issue or answering a specific question. The purpose of a spike is to gather information or to test the feasibility of a solution before committing to its implementation. Learn more about spikes here: Kent Beck, *Extreme Programming Explained: Embrace Change*, 2nd ed. (Addison-Wesley, 2004).

However, being "busy" does not equal creating value for customers and the organization. As a result, you may observe the following issues of an oversized Product Backlog:

- Adding too many items can result in losing meaningful connections between the various ideas, making prioritizing and managing them difficult.
- When the Product Owner clutters a Product Backlog with many ideas, lower-priority items may overshadow high-priority items. This effect can lead to misallocating resources, as the team might inadvertently focus on less valuable features, diminishing the overall value delivered.
- An excessive number of Product Backlog items can increase cognitive workload, making it harder for the team to comprehend the full scope of work. This load can negatively impact decision-making, prioritization, and, ultimately, the team's ability to deliver high-value outcomes.

Remedy: These drawbacks of idea-inflated Product Backlogs typically outweigh the benefits of keeping track of ideas and requirements.

Regularly refining and ordering the Product Backlog, focusing on the most valuable items while maintaining a manageable number of ideas that the team can effectively engage with and deliver, helps teams to avoid the pitfalls of this anti-pattern.

12. Part-Time Product Owner

Observation: The Product Owner is not working daily on the Product Backlog.

See Chapter 2, anti-pattern 1, for a detailed description.

13. Copy & Paste Product Owner

Observation: The Product Owner creates Product Backlog items by breaking down requirement documents received from stakeholders into smaller chunks.

See Chapter 2, anti-pattern 2, for a detailed description.

14. Crowding-Out Collaboration

Observation: The Product Owner invests too much time up front in creating Product Backlog items, making them too detailed.

See Chapter 2, anti-pattern 11, for a detailed description.

15. The Overreaching Product Owner

Observation: The Product Owner creates Product Backlog items by not just providing the why but also defining the what and the how.

See Chapter 2, anti-pattern 4, for a detailed description.

16. The "I Know It All" Product Owner

Observation: The Product Owner does not involve stakeholders or subject matter experts in the refinement process.

See Chapter 2, anti-pattern 12, for a detailed description.

PRODUCT BACKLOG ANTI-PATTERNS OF THE DEVELOPERS

17. Submissive Engineers

Observation: The Developers submissively follow all demands of the Product Owner.

See Chapter 3, anti-pattern 32, for a detailed description.

18. What Technical Debt?

Observation: The Product Backlog does allocate sufficient capacity to maintain the platform, as the Developers do not demand adequate time to tackle technical debt[8] and bugs and to preserve the quality standard of the product. Instead, they fully embrace the feature factory,[9] shipping one new feature after another.

Background: The Developers should consider allocating up to 20 percent of their time to fixing bugs and refactoring the codebase. A high-quality tech stack helps a Scrum Team to adapt to changing conditions and new information.

8. *Technical debt* is a metaphor in software development describing the future work and complexity resulting from short-term solutions over better, long-term alternatives. Accumulating over time, it can lead to increased maintenance costs, slower development, and lower software quality. Learn more: Martin Fowler, "Technical Debt," May 21, 2019, https://martinfowler.com/bliki/TechnicalDebt.html.
9. John Cutler's "feature factory" concept describes a product development environment where teams focus on delivering features rather than creating value for users and the business. In a feature factory, output (the number of features) precedes outcomes (user experience, the desired change in behavior, accomplishing business objectives, etc.). Unfortunately, this misaligned focus can lead to wasted effort and resources. Scrum practices combat the feature factory by emphasizing collaboration, iterative development, and continuous feedback. Cutler introduced this concept in his article "12 Signs You're Working in a Feature Factory," @johncutlefish's blog, November 17, 2016, https://cutle.fish/blog/12-signs-youre-working-in-a-feature-factory.

This time allocation will come at the expense of creating new features and may lead to diverging opinions with the Product Owner. Product Owners may be more interested in short-term feature delivery than the long-term sustainability of the product or service.

This technical debt anti-pattern typically manifests itself in environments characterized by a disconnection between the Scrum Team and the actual users or in scenarios where the immediate pressure for new features overshadows the importance of maintaining a robust and healthy codebase. Venture capital-funded startups, operating with negative cash flow in fierce competition, often fall into this trap. Similarly, large organizations, especially those where production support and development are disjointed or are led by managers with insufficient technical understanding, also tend to undervalue the importance of addressing technical debt, leading to a buildup of long-term issues that compromise the product's sustainability.

There are multiple reasons why Product Owners push for more feature delivery at the expense of the product's or service's long-term sustainability:

- Investors and stakeholders often demand visible progress and regard feature delivery as tangible evidence.
- Fierce competition in the market may drive Product Owners to prioritize feature delivery to stay ahead of or catch up to competitors.
- New features might bring in immediate revenue, which is vital for cash-flow-negative startups to survive.
- Product Owners may face pressure from customers or the sales department to deliver specific features quickly to meet expectations or contractual obligations.
- Some Product Owners might have personal incentives tied to feature delivery, such as bonuses or promotions, which can lead to prioritizing short-term goals over long-term sustainability.
- Product Owners may not fully understand the technical debt and long-term consequences of neglecting the tech stack quality.
- Some Product Owners may underestimate the risks and potential downsides of prioritizing features over long-term sustainability, assuming they can address technical debt later in a case of optimism bias.

Remedy: Technical excellence is a prerequisite of any form of business agility; preserving this state is a continuous process that requires a steady and substantial investment. However, if you ignore the required maintenance work, the decline of your team's capabilities to deliver value is no longer a question of if but when.

I recommend using a Retrospective to kick off the discussion, ultimately addressing the following issues:

- Explain the long-term consequences of neglecting the tech stack, and the importance of addressing technical debt, to the Product Owner.
- Provide quantitative evidence of the impact of technical debt on the team's productivity, product quality, and customer satisfaction.
- Frame the importance of a sustainable tech stack in delivering consistent customer value over time.
- Highlight success stories of other companies benefiting from long-term sustainability investing. Alternatively, share negative examples of organizations failing to address technical debt.
- Communicate with stakeholders to ensure they understand the importance of balancing feature delivery with long-term product health.
- Share experiences and strategies with other teams in the organization facing similar challenges. A community of practice is a great place to have these conversations.
- If the Product Owner remains unresponsive to the team's concerns, consider escalating the issue to higher management for support.

Read more on technical debt and Scrum and who is responsible for what on a Scrum Team.[10]

Product Backlog Anti-Patterns of the Scrum Team

19. Involving the Scrum Team—Why?

Observation: The Product Owner does not involve the entire Scrum Team in Product Backlog refinement and instead relies on just the "lead engineer" and a

10. Stefan Wolpers, "Technical Debt & Scrum: Who Is Responsible?" February 28, 2019, https://age-of-product.com/technical-debt-scrum.

designer. Moreover, the Developers are okay with this approach, as it allows them to write more code which they prefer over figuring out what is worth building.

See Chapter 2, anti-pattern 15, for a detailed description.

20. No Time for Refinement

Observation: The Developers do not have enough Product Backlog refinement sessions, resulting in a low-quality backlog.

See Chapter 3, anti-pattern 29, for a detailed description.

21. Too Much Refinement

Observation: The Scrum Team has too many refinement sessions, resulting in a too-detailed Product Backlog.

Background: Too much refinement isn't healthy, either. There is a moment when the marginal return of an additional refinement effort is zero or possibly even negative—think analysis paralysis.

Remedy: The only way to understand whether the previous validation of a new feature, for example, your prototypes, is correct is to build and ship it. There is no way for the Scrum Team to figure this out at the green table, refining and discussing the issue endlessly: if you don't ship it and gather feedback from real customers to inform your next steps, you kill the idea and move on.

FOOD FOR THOUGHT

Is a Product Backlog, per se, a waste? Some lean practitioners consider the Product Backlog artifact already a waste, as they would prefer just-in-time delivery of valuable Product Backlog items at the beginning of the Sprint Planning.[11]

Product Owner vs. product manager: Do we need a product manager in an organization dedicated to practicing Scrum? Often, particularly in larger organizations, we notice a separation between these two roles: The Product Owner is a tactical job

11. Allan Kelly, "Backlogs, #NoBacklog(s) and Comfort Blankets," September 2022, https://www.allankelly.net/archives/6541/backlogs-nobacklogs-and-comfort-blankets.

at the team level, delivering what the product manager created at the organizational level. However, is this approach a beneficial division of responsibilities? After all, according to the process described earlier, the Product Owner is also responsible for product discovery.

Conclusion

Even if you have successfully identified what is worth building next, you probably need to improve your Product Backlog and your Product Backlog refinement process. Bring this issue up with your team in your next Retrospective and consider possible Product Backlog anti-patterns.

Addressing some of the issues in this chapter is the easiest way to improve your team's performance and thus the team's standing among stakeholders and customers. Solving Product Goal and Product Backlog anti-patterns offers the best return on investment from a Scrum perspective. After all, the outcome of your Scrum Team's work is only as good as the input, your Product Backlog.

Toolbox

Regarding the following tips and exercises, you will find a detailed description in Appendix B:

- **"Creating Product Goals as a Collaborative Practice"** suggests using Ralph Jocham's Product Goal Canvas to create a shared understanding among the Scrum Team members about the why, what, and how of the team's journey.

This page intentionally left blank

11 Sprint Backlog Anti-Patterns

Introduction

While the Product Backlog has a simple structure, it contains the work the Scrum Team deems necessary to accomplish the Product Goal at any given moment in time; the Sprint Backlog has a more challenging nature; see Figure 11.1.

"The Sprint Backlog is composed of the Sprint Goal (why), the set of Product Backlog items selected for the Sprint (what), as well as an actionable plan for delivering the Increment (how)."[1]

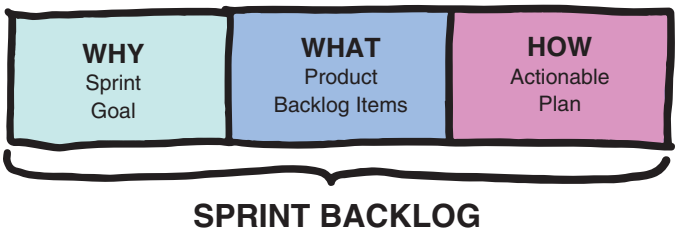

Figure 11.1 The Sprint Backlog concept.

1. Ken Schwaber and Jeff Sutherland, "Sprint Backlog," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#sprint-backlog.

Compared to the Sprint Backlog, the Product Backlog does not comprise the Product Goal or the plan to accomplish it; it is an "emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team."[2]

Scrum Teams should notice this difference between the two Backlogs. Many practitioners assume that the Sprint Backlog is merely a subset of the Product Backlog where the Product Owner hands over responsibility for its composition to the Developers. Moreover, there is additional confusion resulting from different time spans. While the Product Backlog exists as long as the Scrum Team builds the product, a Sprint Backlog has a temporary nature: at the end of the Sprint, it will be gone. Consequently, the Sprint Backlog's time horizon is significantly shorter, no longer than a month. This reduced time span requires significant work in creating and maintaining an actionable plan to accomplish the Sprint Goal.

Conversely, Scrum Teams typically do not have a detailed plan regarding the Product Goal. Given the uncertainty surrounding the long-term Product Goal, creating it would likely be a wasteful enterprise. So instead, they treat the Product Goal as a hypothesis and validate it one Sprint at a time, willing to change direction based on data and feedback from stakeholders and the marketplace.

As we will see, these differences are crucial to understanding the many ways Scrum Teams can misapply the concept of the Sprint Backlog.

SPRINT BACKLOG ANTI-PATTERNS

Given the interdependencies between Sprint Planning, Sprint Goal, and the resulting Sprint Backlog, Sprint Backlog anti-patterns often originate with the former two. For example, failure of the Scrum Team to adequately estimate their available capacity in the upcoming Sprint may result in their taking on an overly optimistic Sprint Goal, leading to undone work at the end of the Sprint, possible failure to achieve the Sprint Goal, and possible failure to produce valuable, usable product Increments.

2. Ken Schwaber and Jeff Sutherland, "Product Backlog," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#product-backlog.

Sprint Backlog Anti-Patterns of the Scrum Team

1. Not Having a Sprint Backlog by Skipping the Sprint Goal

Observation: The Scrum Team cannot decide on a Sprint Goal. Consequently, they choose to work on a list of noncohesive, arbitrarily chosen Product Backlog items and never create a Sprint Backlog.

Background: The Sprint Goal is an essential part of the Sprint Backlog; for example:

- It defines why the Sprint is valuable.
- It informs the Scrum Team which Product Backlog items to choose to accomplish the Sprint Goal.
- It allows the Scrum Team to work as a cohesive unit.
- It enables the team to measure whether they accomplished the goal at the end of the Sprint.

Technically, not having a Sprint Goal prevents the creation of a formal Sprint Backlog. Consequently, the lack of a Sprint Goal often results in the team not working as a cohesive unit but as individuals, toiling on disparate tasks. In practice, this phenomenon often leads to a team focused on creating output—in this case, releasing as many Product Backlog items per Sprint as possible. Instead, from a Scrum perspective, the team should focus on maximizing the outcome of a Sprint, thus creating value for customers.

Remedy: During Sprint Planning, focus as a Scrum Team on creating a Sprint Goal that is achievable by the end of the Sprint and contributes to the overall Product Goal. If that is impossible, analyze the root cause for the failure. (To learn more on this topic, please see Chapter 14, anti-pattern 4.)

2. The Sprint Goal Is Overly Ambitious

Observation: The Scrum Team decides to work toward an overly ambitious Sprint Goal, which leads to a bloated Sprint Backlog. Consequently, the Scrum Team fails to deliver a done product Increment at the end of the Sprint.

See Chapter 14, anti-pattern 3, for a detailed description.

3. The Maverick and the Sprint Backlog

Observation: Someone adds items to the Sprint Backlog without consulting the Developers first.

Background: Having Developers control their own Sprint Backlog helps them forecast. They may add to the Sprint Backlog, for example, when they discover they need to fix a critical bug after the Sprint's start or when they identify new work that is vital to achieving their Sprint Goal. The Developers need to be the ones to decide they will take on this work in the Sprint, because they often have to take other work off the Sprint Backlog to make room for the new work. Only the Developers have the knowledge to make these decisions, and the decisions are theirs alone.

Remedy: No stakeholder or individual Scrum Team member can add or remove an item to or from the Sprint Backlog without consulting the Developers, who have a veto right. If stakeholders continue doing so, reach out to them to raise their awareness of how damaging this behavior is to the team's self-managing, thus putting the accomplishment of Sprint Goals at risk. If a team member is doing so, address the issue at the next Retrospective and conclude how to avoid this in the future.

> ***Practical Tales:*** *I worked with two Scrum Teams on a B2C application in a very competitive market. One Sprint, we worked hard to successfully reduce the loading time of the homepage only to see the advantages wholly rolled back in the following Sprint. In the beginning, we were clueless until the most senior developer mentioned during a coffee break that he was disappointed about the real-time performance of a brand-new JavaScript framework. Over the weekend, he integrated that framework into our application to run a "test with real traffic" without considering it necessary to tell anyone.*

4. No Visualization

Observation: The Scrum Team does not visualize their progress toward their Sprint Goal. As a result, the Sprint Backlog remains an undifferentiated pile of work items throughout the Sprint, reducing transparency.

Background: Given its tactical nature as a temporary artifact, a Scrum Team is well advised to bring maximum transparency at the work item level to the Sprint Backlog. From a Scrum perspective, a Product Backlog item is done or not done.

From a team perspective, to monitor the progress toward the Sprint Goal, this binary approach is insufficient.

Its daily analysis during the Daily Scrum supports one purpose: Is the Scrum Team on track to meet the Sprint Goal? The analysis covers a lot of tactical planning; for example:

- Review progress on individual work items.
- Assess remaining effort.
- Check work item dependencies.
- Identify potential impediments.
- Reconsider previous work item allocation.

If the Developers do not understand where they are concerning progress toward the Sprint Goal, it becomes less likely that they will achieve it. Successful Sprints result from increasing Developers' confidence over time. They do not materialize miraculously on the last day of the Sprint.

The Scrum Guide does not mention that Scrum Teams should use visualization in general or Sprint boards in particular to visualize the Sprint Backlog. However, choosing a practice such as Kanban[3] to support the Developers by representing the actual state of the planned work is highly beneficial: a Sprint board provides transparency and points at possible problems at an early stage—for example, a work item that is not moving toward done.

Remedy: Visualizing the Sprint Backlog and thus the progress toward the Sprint Goal is useful. Several options exist for accomplishing this Sprint progress visualization, from Sprint boards in various forms to burndown charts. If the

3. A Kanban board can help the Scrum Team visualize their progress toward the Sprint Goal by clearly representing the workflow, tasks, and statuses. The Scrum Team typically divides the board into columns, each representing a stage in the workflow, such as To Do, In Progress, and Done. Then, team members move tasks, represented by cards, across the board as they work on them and complete them. This practice allows the Scrum Team to easily track the progress of each task and identify bottlenecks, potential impediments, and areas that may require additional attention. Using a Kanban board, the Scrum Team can maintain a shared understanding of their progress and ensure they are on track to meet the Sprint Goal. In addition, it helps increase transparency, fosters collaboration, and enables the team to quickly adapt and reprioritize tasks as needed. Learn more: David J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business* (Blue Hole Press, 2010).

Developers struggle to visualize the progress toward the Sprint Goal, consider running a workshop on Sprint boards.[4]

SPRINT BACKLOG ANTI-PATTERNS OF THE DEVELOPERS

5. Planning Too Detailed

Observation: During the Sprint Planning, the Developers plan every single task of the upcoming Sprint.

Background: By turning the Sprint Backlog into a comprehensive plan, the Developers lock themselves into the solution that they have in mind at the time. They close off the possibility that they may learn things during the Sprint that help them devise a better solution. This learning is often essential to achieving Sprint Goals in complex environments. Scrum dedicates an event, the Daily Scrum, to facilitate these learning opportunities.

When Developers create a comprehensive plan, they demonstrate that they have an illusion of control that makes it harder for them to adapt to unexpected information or events. Comprehensive plans are usually an indication of the lack of an effective agile mindset.

Remedy: Developers should not become too granular in their planning during Sprint Planning. For example, planning two to three days ahead for a two-week Sprint is more than sufficient to start with the Sprint and start learning. The Sprint Backlog is emergent; too much planning up front might result in waste. Scrum is not a timeboxed form of the waterfall planning practice.

6. Too Much Estimating

Observation: During the creation of the Sprint plan, the Developers estimate Product Backlog items from the Sprint Backlog down to the level of subtasks.

Background: Many people in the Agile community generally question allocating time to forecasting the effort to deliver a done Increment. Instead, they suggest merely counting the number of tasks and skipping detailed estimating—what they consider waste. They argue that over time a Scrum Team gains a good understanding of the

4. Stefan Wolpers, "Create Whiteboards for Transparency and Collaboration," June 6, 2018, https://age-of-product.com/create-whiteboards.

number of Product Backlog items they can handle per Sprint. While their productivity will vary from Sprint to Sprint within certain boundaries, the variations do not merit the additional work that goes into the detailed estimating of items. The team should allocate this effort instead to work from the Sprint Backlog.

However, besides coming up with a numerical forecast, the real advantage of estimating is something different: it can help to create alignment among Developers on the how and who of the tasks from the Sprint Backlog.[5]

Remedy: Estimating subtasks looks to me like accounting for the sake of accounting. Instead, as a team, identify the sweet spot where the return on estimating becomes negative, and restrain yourself from going beyond it. Read more: "Estimates Are Useful, Just Ditch the Numbers."[6]

7. Gold-Plating

Observation: The Developers increase the scope of the Sprint beyond the Sprint Goal by adding unnecessary work to the Sprint Backlog.

See Chapter 3, anti-pattern 5, for a detailed description.

8. Cherry-Picking Product Backlog Items Unrelated to the Sprint Goal

Observation: The Developers choose Product Backlog items totally unrelated to the Sprint Goal for the Sprint Backlog. Consequently, the Sprint Backlog resembles a random assortment of work items based on preference, not the Sprint Goal.

See Chapter 14, anti-pattern 11, for a detailed description.

9. Fixed Scope

Observation: The Sprint Backlog's scope is fixed throughout the Sprint. The Developers do not add new work items or remove unnecessary work items from the Sprint Backlog.

---

5. Beware, though; there is always a moment when adding more effort to estimate tasks or improve the quality of existing estimates will no longer be an advantage from the perspective of risk mitigation or Sprint Goal accomplishment. Beyond that point, spending more time estimating will no longer deliver a return on investment for the Scrum Team. This issue is a good topic for a Retrospective on your team's estimation practices: Have we crossed the point where additional estimating delivers negative returns?
6. Stefan Wolpers, "Estimates Are Useful, Just Ditch the Numbers," September 20, 2021, https://age-of-product.com/estimates-useful-ditch-numbers.

Background: The Sprint Backlog is emergent: "If the work turns out to be different than they expected, they collaborate with the Product Owner to negotiate the scope of the Sprint Backlog within the Sprint without affecting the Sprint Goal."[7]

Changing the Sprint Backlog is possible, as the Sprint Goal shall provide some wiggle room for flexible responses to emerging challenges. For example:

- The Developers discover new necessary work only during the Sprint.
- They run into unexpected technical issues that require them to change their approach to meeting the Sprint Goal.
- Some Product Backlog items deemed previously necessary are not required: Why work on those if they do not contribute to accomplishing the Sprint Goal?

Moreover, what about an urgent bug that appears suddenly? If the payment function of your e-commerce site is no longer working, you will fix this immediately—no matter what the Sprint Goal says.

Remedy: Scrum is not about following the initial Sprint plan to the letter. Scrum is about adapting to lessons learned during the Sprint to deliver the Sprint Goal. This approach of inspecting progress toward the Sprint Goal and adapting the plan is the rationale behind dedicating the Daily Scrum to facilitate these learning opportunities at least once daily. If Developers struggle with the approach, try to learn why that is the case:

- Maybe the organization's culture treats forecasts as a commitment and rewards accomplishing them.
- It could also be that the same culture penalizes a failure to deliver what the Developers forecasted.
- Alternatively, the Developers may not be interested in accomplishing the Sprint Goal. However, they appreciate a smooth ride by fixing the scope of their work during the Sprint at the beginning.

In this case, a Retrospective may be a good starting point, bolstered with insights from an anonymous survey and one-to-one interviews.

7. Ken Schwaber and Jeff Sutherland, "Sprint Backlog," *The 2020 Scrum Guide*,™ https://scrumguides.org/scrum-guide.html#sprint-backlog.

10. The Urge to Deliver the Complete Sprint Backlog

Observation: The Developers feel obligated to finish all work items of the Sprint Backlog no matter whether or not those contribute to accomplishing the Sprint Goal.

Background: The Sprint's purpose is to deliver the Sprint Goal that is most valuable to the customers and the organization at that very moment.

Consequently, the Sprint is not about providing as many done work items as possible or keeping the Developers at a 100 percent utilization rate. In this context, if the Scrum Team delivers the Sprint Goal, it is, by all means, acceptable to end the Sprint with some undone Product Backlog items in the Sprint Backlog.

Remedy: This Sprint Backlog anti-pattern partly overlaps with the previous "Fixed Scope" anti-pattern. Again, try to identify the motivation's origin to clear the Sprint Backlog at the end of the Sprint. There could be several reasons for that behavior; for example:

- The organization values a high utilization rate of workers, and managers push to meet expectations. (Maybe, it is an artifact from its [industrial] past?)
- The organization rewards when Scrum Teams deliver "commitments" at the end of the Sprint.

Again, a Retrospective may be a good starting point, bolstered with insights from an anonymous survey and one-to-one interviews. Once you understand the issue better from the Scrum Team's perspective, head out and repeat the exercise with the stakeholders. Finally, bring the two parties together and help them reach a better agreement that embraces trust and self-management, as the latter is beneficial to the organization, too. Read more: "Self-Management—The Top Ten Business Reasons to Trust Your Teams."[8]

8. Stefan Wolpers, "Self-Management—The Top Ten Business Reasons to Trust Your Teams," April 4, 2023, https://age-of-product.com/self-management.

Practical Tales: *I worked at a prominent startup where the founders kept track of the Scrum Teams' velocities and the forecasted vs. done work ratio. During every Sprint Planning, they pointed to commitment gaps of the teams based on their understanding of the possible delivery scope. Also, at the end of the Sprint, they demanded a detailed report on work items forecasted and delivered. Unfortunately, we could never convince them of the benefits of not controlling their teams but instead trusting them. The problem solved itself when Google acquired the startup.*

11. The Shadow Sprint Backlog

Observation: Not all the work of the current Sprint is available in the Sprint Backlog because the Developers are also working on other tasks, compromising their focus on accomplishing the Sprint Goal.

Background: There are various reasons for unaccounted work during a Sprint; for example:

- A stakeholder or manager asks a Developer to support another team temporarily, and they do not add these unrelated tasks to the Sprint Backlog.
- The Developers do not add newly identified (necessary) work to the Sprint Backlog, as they believe that the Sprint scope is fixed.
- They try to avoid making it transparent that their original forecast was off. (This is a common effect in a complex environment; not accepting it may indicate a lack of psychological safety.)
- The Developers forget to do so or decide not to do so as they consider the resulting accounting effort to be wasteful.

The worrying part of this anti-pattern is the reduced level of transparency. How are the team members supposed to inspect the progress toward the Sprint Goal and change their plan if only incomplete information on the current state is available to them?

You are likely to experience this problem, as transparency is a tricky field. On one hand, we want to be as transparent as possible to understand our progress toward the Sprint Goal, which calls for comprehensive task accounting. On the other hand,

we sincerely believe in self-management and the professionalism of our teammates; a Scrum Team does not resemble a surveillance state. Consequently, you will only learn to balance these two sides over time.

Remedy: There are several ways to address the issue:

- If stakeholders cause the issue, reach out to them and help them understand how their behavior may compromise a Scrum Team's ability to deliver its Sprint Goals.
- If it is a team-internal issue, determine what is causing this behavior: ignorance, laziness, or a lack of psychological safety?

While the team can address ignorance and laziness in a Retrospective, tackling a lack of psychological safety is more challenging and beyond the scope of this book.

SPRINT BACKLOG ANTI-PATTERNS OF THE PRODUCT OWNER

12. Additional Work

Observation: The Product Owner or even other stakeholders attempt to introduce new work to the current Sprint, for example, during the Daily Scrum or by reaching out to individual Developers; see Figure 11.2.

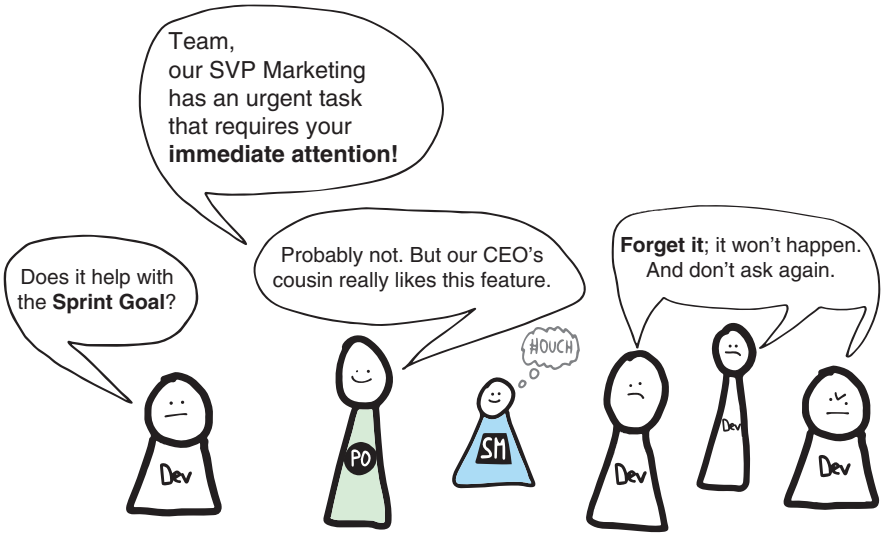

Figure 11.2 If new work emerges during the Sprint that does not contribute to the Sprint Goal, the Scrum Team needs to negotiate whether or not to take it on. Life is a negotiation; why would Scrum be different?

Background: While the Sprint Backlog is not fixed, it is related to the scope of the Sprint Goal. The scope may be subject to negotiation between the Product Owner and the Developers if the Scrum Team learns something during the Sprint that affects the original Sprint Goal. However, the team members and stakeholders should not misuse Scrum's built-in flexibility to generally challenge the Sprint Goal or stuff the Sprint by simply adding more work arbitrarily.

Adding more work, probably even work unrelated to the Sprint Goal, may be acceptable for high-priority bugs. But, admittedly, the Developers should address those immediately once they learn about them.

Remedy: The composition of the Sprint Backlog is the responsibility of the Developers: they may accept new work during the Sprint. However, that is their sole decision, as Scrum is a pull system, not a push system.

If your stakeholders are unaware of this fact or prefer to ignore it, reach out to them to help them understand how their behavior is putting accomplishing Sprint Goals at risk.

Also, it does not hurt to prepare for the unforeseen by adding some buffer to the capacity planning part of Sprint Planning. Depending on the context of a Scrum Team, reserving up to 10 percent buffer time is a cautionary step.

13. Absent Product Owner

Observation: The Product Owner is absent for most of the Sprint and is not available to answer questions from the Developers regarding Product Backlog items from the Sprint Backlog.

See Chapter 2, anti-pattern 20, for a detailed description.

14. Product Owner Changes Work Items from the Sprint Backlog during the Sprint

Observation: The Product Owner cannot let go of Product Backlog items once they become part of the Sprint Backlog. For example, the Product Owner increases the scope of a work item or changes acceptance criteria once the Developers select this Product Backlog item for the Sprint Backlog; see Figure 11.3.

Background: There is a clear line: the Product Owner is responsible before a Product Backlog item becomes part of the Sprint Backlog. However, once it moves

from one backlog to the other, the Developers become responsible. There is no sneaking in additional work through the backdoor, as it violates the prerogative of the Developers to decide on how to turn a Product Backlog item into a done Increment. Also, it is disrespectful and undermines trust among the team members.

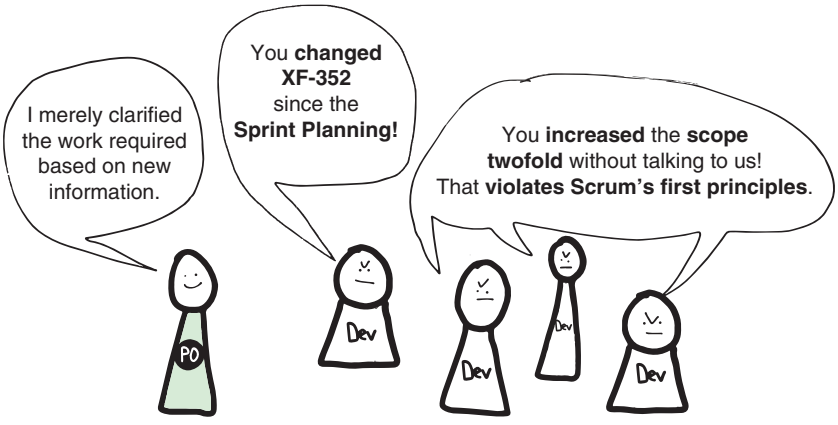
I merely clarified the work required based on new information.
You **changed XF-352** since the **Sprint Planning!**
You **increased** the **scope twofold** without talking to us! That **violates Scrum's first principles**.
PO
Dev
Dev
Dev
Dev

Figure 11.3 Without the Developers' consent, no one can touch Product Backlog items that are part of the Sprint Backlog.

Remedy: If changes become acute during the Sprint, the team will decide how to handle them collaboratively. If the Product Owner cannot restrain themselves, take it to the next Retrospective.

15. Sprint Stuffing

Observation: The Developers accomplished the Sprint Goal early, and the Product Owner is pushing them hard to accept new work from the Product Backlog to fill the void of the Sprint Backlog.

Background: The Scrum Team decided collaboratively on the Sprint Goal, and the Developers committed to delivering. Consequently, it is the prerogative of the Developers to decide on the composition of the Sprint Backlog. Should they manage to accomplish the Sprint Goal before the Sprint's end, it is their sole decision to fill the remaining time, for example, by

- Accepting new work from the Product Backlog,
- Refactoring parts of the tech stack,
- Exploring new technology that might be useful,

- Fixing some bugs, and
- Sharing knowledge with fellow teammates.

Remedy: Scrum is not in the business of maximizing the utilization rates of team members. Achieving the Sprint Goal is sufficient, as it is the most valuable addition to the product the team can accomplish during the Sprint.

However, should the team regularly deliver the Sprint Goal early, the Scrum Team may want to inspect that behavior during a Retrospective. The team may be playing safe, whatever the reason. In this case, the urge of the Product Owner to stuff the Sprint Backlog is less of an anti-pattern and more of an expression that the team needs to address the elephant in the room.

Food for Thought

Pursuing their agenda: The Developers are accountable for creating a Sprint Backlog. What if they use that privilege to focus on creating what they consider essential, disguising it as necessary Sprint work?

The predefined Sprint Backlog: What if a Scrum Team practices continuous Product Backlog refinement at a level that creates an actionable Product Backlog, allowing the team to run a Sprint Planning at a moment's notice? In that situation, ordering the Product Backlog based on the previous discussions and plans already represents a kind of Sprint Backlog for the next two or three Sprints. Would it be necessary to have a formal Sprint Planning event in that situation, creating the "official" Sprint Backlog?

Conclusion

The Sprint Backlog reflects the Sprint Goal and thus is a subset of the overarching Product Goal and business objectives at a tactical level. Next to work items, it comprises the Sprint Goal and the Developers' plan to accomplish it. It is at the heart of Scrum's checks and balances, putting the Sprint Backlog firmly into the hands of the Developers.

No matter how well versed your Scrum Team is in practice, no one on the side of the stakeholders will care unless the Scrum Team regularly delivers Increments every Sprint. Consequently, the Sprint Backlog is a mission-critical artifact for any Scrum Team.

12 Increment Anti-Patterns

Introduction

Ultimately, Scrum is a framework that enables iterative and incremental product development in a complex environment by delivering in short bursts—called Sprints—to mitigate the risk of moving in the wrong direction. However, avoiding building something no user or customer wants requires closing the learning loop as frequently as possible.

Consequently, the only way to create value is to ship Increments to the real users and listen to their feedback; just building a "potentially shippable" Increment does not do the trick. Anything that the Scrum Team does not release is just work of unknown value.

Whenever possible, Scrum Teams should release their Increments to customers to obtain direct feedback. Cloud-based environments allow Scrum Teams to release many times a day, enabling Scrum Teams to obtain rapid feedback and learn quickly.

Admittedly, there are also situations where this is more difficult to achieve; for example:

- The market is highly regulated, and releases must be approved by a governance body.

- Customers can't absorb frequent releases because of potential changes releases may cause to their business processes. They need time to adapt.
- It is impossible to create a valuable Increment in a short period. For example, training large language models does not work in a few days.

In addition, releasing Increments to real users represents not just the closing of the learning loop but probably also signals that the Sprint has ended and a new Sprint can start. In its own way, releasing Increments is a form of gratification for all team members, a reason to celebrate. (This effect may wear off when you practice continuous delivery.)

We cannot overemphasize the importance of releasing Increments, or, as Ron Jeffries puts it: "I have one goal: to describe why the Increment is the most important Scrum element for developer success and survival."[1]

The Purpose of the Increment According to the Scrum Guide

The Scrum Guide characterizes Increments as follows:[2]

1. Imagine the current Product Goal your Scrum Team is pursuing as a puzzle. Then, all the Increments required for the Product Goal represent individual pieces of that puzzle. Individually, they may be of little value. However, in their entirety, they create the picture. Increments are inherently additive; they build on each other. (Of course, for the sake of simplicity of this metaphor, we assume that the Product Goal is valuable.)
2. We consider every Product Backlog item that meets the Definition of Done during a Sprint an Increment. Therefore, we may create multiple Increments during a Sprint.
3. The Scrum Team decides when to deliver Increments to stakeholders; there are no release gates in Scrum. There is no obligation
   a. To deliver any Increment at all during the Sprint, or
   b. To deliver only at the end of the Sprint.

1. Ron Jeffries, "Increment? What's That?" June 24, 2021, https://ronjeffries.com/articles/021-01ff/Increment.
2. Ken Schwaber and Jeff Sutherland, "Increment," *The 2020 Scrum Guide™*, https://scrumguides.org/scrum-guide.html#increment.

4. Scrum Teams deliver Increments whenever they meet the Definition of Done, and the team deems the delivery beneficial for the stakeholders. In other words, Scrum and continuous delivery (CD)[3] work well together.

Whether you like it or not, your stakeholders will judge your Scrum Team by one achievement: Can your team deliver valuable Increments every single Sprint? Given the importance of this practice, you invest time well if you consider Increment anti-patterns that prevent your team from meeting this expectation.

INCREMENT ANTI-PATTERNS

INCREMENT ANTI-PATTERNS BY THE SCRUM TEAM

1. No Decision on the Release of Increments

Observation: Although the Scrum Team creates a potentially valuable Increment, the team cannot agree to release them. Therefore, the team does not ship anything.

Background: The Scrum Team does not create any value without releasing Increments.

Not releasing Increments also means not closing the feedback loop with actual users of the Increments. The resulting missed learning opportunity immediately affects the empirical process Scrum uses. It increases the risk that the team cannot inspect and adapt its assumptions of what is valuable to your users or customers, reflected by the Product Goal and the composition and ordering of the Product Backlog. The longer the Scrum Team decides to hold back releases, the higher the probability it may pursue the wrong path to creating value.

Some reasons why a Scrum Team may not release an Increment are as follows:

- **Dependency on other teams or systems:** The Increment may depend on another Scrum Team's work or an external system that is not yet ready or integrated.

3. Continuous delivery is a software development practice that automates and streamlines releasing software updates and bug fixes, enabling teams to release new code into production with minimal manual intervention reliably. This approach allows teams to deliver features, updates, and fixes more frequently and with less risk, enhancing software quality and responsiveness. Learn more: Jez Humble and David Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation* (Addison-Wesley, 2010).

- **Organizational policies:** Some organizations have strict, Scrum-defying guidelines around release schedules or specific release windows during which a Scrum Team can release new features.
- **Stakeholder concerns:** Occasionally, stakeholders might have issues or want to delay the release due to market conditions, customer feedback, or other external factors and convince the Product Owner that a delay is in the best interests of everyone.
- **Technical challenges:** Sometimes, a Scrum Team might face technical challenges or constraints that prevent them from releasing the Increment, such as issues with deployment, infrastructure, or performance optimization.
- **Awaiting further features:** The Scrum Team may decide to bundle multiple Increments together in a more significant release, particularly if the current Increment relies on upcoming features or improvements for optimal user experience.

Remedy: Ultimately, the only way for a Scrum Team to create value is to ship Increments to the real users and listen to their feedback. Anything below that threshold is waste.

If a Scrum Team is indecisive or unmotivated to release, I strongly recommend addressing this issue in one of the subsequent Retrospectives; it should rank high on the Scrum Team's list of planned improvements.

2. Postponed Releases

Observation: The Scrum Team releases Increments only at the end of the Sprint.

Background: As the Scrum Guide states: "However, an Increment may be delivered to stakeholders prior to the end of the Sprint."[4] Technically, whenever a Product Backlog item meets the Definition of Done, the Scrum Team can decide to release it. At least as far as Scrum is concerned, there is neither an internal Scrum Team approval gate nor one by stakeholders.

Notably, the idea of a release gate does not apply to the Sprint Review. Here, the Scrum Team and stakeholders inspect the Increments from the recent Sprint and thus progress toward the Product Goal and adapt the Product Backlog if necessary.

4. Ken Schwaber and Jeff Sutherland, "Sprint Backlog," *The 2020 Scrum Guide*™, https://scrumguides.org/scrum-guide.html#increment.

However, there is no sign-off of Increments by stakeholders. That ship has already sailed.

Remedy: If the Scrum Team considers an Increment both done and valuable, it can release the Increment so long as customers can adopt it and so long as it meets governance rules and stakeholders' preferences. The sooner the team releases the Increments, the sooner they will be able to obtain feedback, helping them to lower their risk of moving in a direction that diminishes value creation.

3. The Scrum Team Only Releases to the Complete User Base

Observation: The Scrum Team always ships all Increments to all users.

Background: The Scrum Guide does not mention that an Increment must be available to all users or customers once released. No rule excludes shipping Increments to specific user groups, for example, by using feature toggles[5] or by selectively releasing only to certain environments.

Remedy: Use all deployment tactics[6] to support your team's efforts in product discovery. Releasing new Increments to a subset of your user base is a powerful way of doing so.

However, ensure that this approach does not undermine the Definition of Done but allows for the preservation of product quality in the long run. Known problems are, for example,

- **Technical debt:** Failing to remove feature toggles at the end of a test phase is a known way of contributing to accumulating technical debt.[7]
- **Merge conflicts:** As different parts of the codebase have various feature toggles enabled, Developers may face merge conflicts when trying to integrate their changes.

5. A feature flag, or feature toggle, is a software development technique enabling developers to turn specific features on or off using conditional statements in the code. This allows testing new functionality, performing A/B testing, and controlling features without code changes or redeployments. Learn more: Wikipedia, "Feature Toggle," updated August 7, 2023, https://en.wikipedia.org/wiki/Feature_toggle.
6. Learn more about software deployment tactics from Humble and Farley, *Continuous Delivery*.
7. Technical debt is a metaphor in software development describing the future work and complexity resulting from short-term solutions over better, long-term alternatives. Accumulating over time, it can lead to increased maintenance costs, slower development, and lower software quality. Learn more: Martin Fowler, "Technical Debt," May 21, 2019, https://martinfowler.com/bliki/TechnicalDebt.html.

- **Testing challenges:** Testing all possible combinations of feature toggles can become complex and time-consuming, leading to inadequate testing coverage.
- **Inconsistent user experience:** Deploying features to only a portion of the user base can lead to a varying user experience, causing confusion and frustration for users and impacting the overall perception of the product.
- **Increased maintenance overhead:** Managing feature toggles requires additional effort to enable or disable features, monitor their usage, and eventually remove them when they are no longer needed.

4. Releasing an Increment Means Handing It Over to the QA Department or Operations

Observation: The Scrum Team does not consider itself accountable for releasing the Increment, as there is a quality assurance department or a committee acting as a release gate outside the Scrum Team, making release decisions; see Figure 12.1.

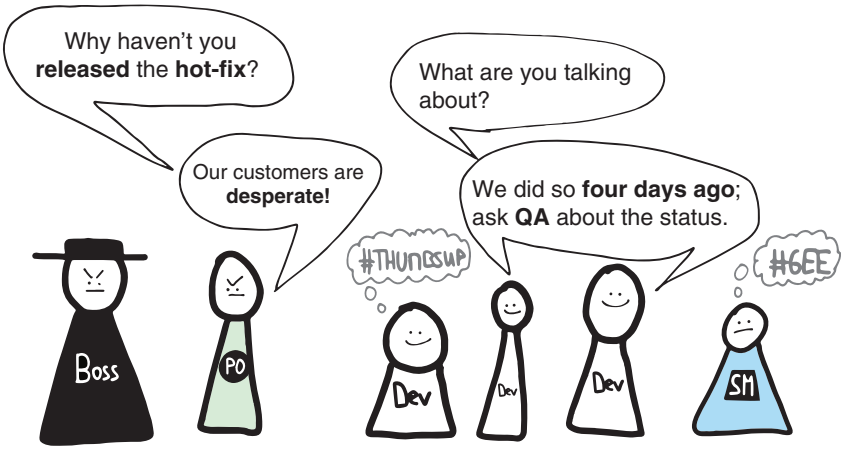

Figure 12.1 In Scrum, there is no quality assurance entity outside the Scrum Team. On the contrary, the team is collectively responsible for quality and release, whereas releasable always entails delivery to the customers' active, live system without repercussions.

Background: This Increment anti-pattern is a sign of an organizational design that prevents a Scrum Team from being accountable for their results. Ultimately, the Scrum Team is responsible for the Increment's quality and delivery to stakeholders. When the Scrum Team is not responsible for release decisions, they are deprived of being accountable for their work. They are incapable of self-management and have become a mere delivery agency.

Remedy: Scrum Team members who cover testing and quality assurance are best able to understand when a Product Backlog item meets the Definition of Done. Separating quality assurance from the Scrum Team reduces autonomy and accountability and creates unnecessary queues and communication overhead that impede value creation. Responsibility for quality assurance should be placed where it delivers the most value: in the Scrum Team.

5. The Scrum Team Releases "Increments" Not Meeting the Definition of Done

Observation: The Scrum Team decides to release an undone piece due to pressure from stakeholders; see Figure 12.2.

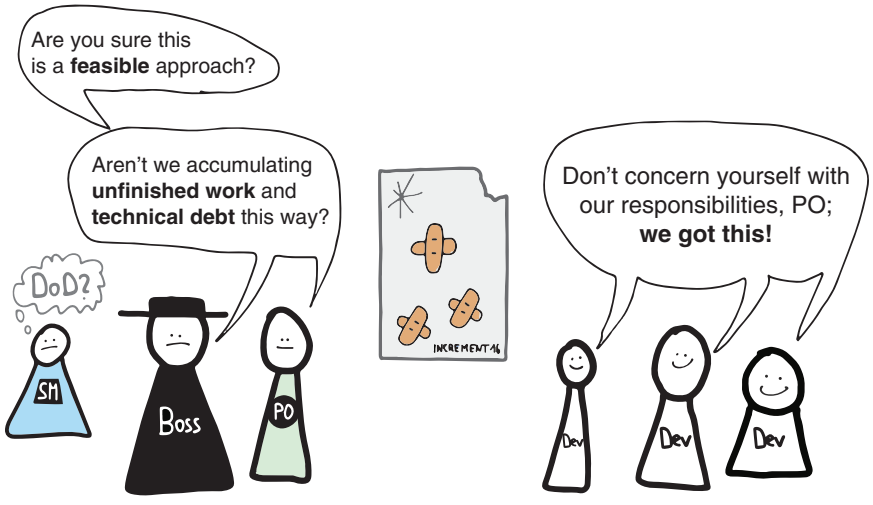

Figure 12.2 There is no corner-cutting in Scrum. As a result, we do not deliver undone work, as the organization's and the Scrum Team's long-term sustainability rest on a high-quality technology stack adhering to a transparent quality standard.

Background: The Scrum Guide is unambiguous in this case: "The Definition of Done is a formal description of the state of the Increment when it meets the quality measures required for the product."[8] Any work that does not meet the Definition of Done cannot be released or addressed at the Sprint Review.

8. Schwaber and Sutherland, "Increment."

Case closed? No, there might be exceptions to the rule:

- Imagine an urgent bug fix in a life-or-death situation. Here, clinging to the sacred principle that all Increments must adhere to the Definition of Done may be considered dogmatic, causing unintended consequences.
- Or imagine the need to build a throwaway prototype that uses actual data. Building it to the usual quality standard would defy its purpose in the first place: figuring out whether it is worth making it, meeting the Definition of Done.
- Finally, imagine your startup is running out of cash. To secure the next funding round, you must deliver a milestone to your investors. What if your remaining runway would allow you to provide that milestone only if you disregarded your Definition of Done? Would you go out of business but uphold a vital principle? Or would you take a fighting chance to save the startup and all jobs it provides?

Remedy: There may be moments when a deviation from the rule is justified; just beware of two essential conditions for doing so:

1. Ensure that everyone understands that this act is the exception and not the new rule.
2. As soon as the critical moment has been mastered, immediately "fix" the accrued technical liability.

> ***Practical Tales:*** *When I worked for a prominent startup, the notion of "no one cares for your Definition of Done when we go out of business" was a regular theme in discussions with stakeholders, particularly the founders. Operating in a dynamic market with better-funded competitors, they valued short-term delivery of features to stay in the race much higher than our long-term perspective of preserving the quality of the technology stack. Despite all the efforts, we never managed to identify a solution satisfying both factions. Finally, the problem solved itself when a popular search engine acquired the startup and introduced its own technical platforms.*

6. The Product Increment Fails to Achieve the Desired Level of Quality

Observation: The Definition of Done is not adequate to meet the expectations of stakeholders and customers. Instead, bugs slip through to production, and trust in the Scrum Team's capabilities deteriorates.

See Chapter 15, anti-pattern 7, for a detailed description.

7. Cutting Corners to Meet an Increment's Arbitrary Deadline

Observation: The Developers reduce product quality below the level defined in the Definition of Done to meet an imposed release deadline.

See Chapter 3, anti-pattern 6, for a detailed description.

8. Delivering Y instead of X

Observation: The Product Owner believes they are getting X. The Developers are working on Y.

Background: This is not merely a result of an inferior Product Backlog item. This anti-pattern indicates that the Scrum Team failed to create a shared understanding about what would be valuable to their stakeholders. There are plenty of reasons for this to happen; for example:

- The Product Owner and the Developers are not talking enough during the Sprint.
- The Product Owner is too busy to answer the team's questions or attend the Daily Scrum.
- Product Backlog refinement sessions during the Sprint are inadequate.
- Developers don't participate in conversations with customers or users, so they lack understanding of the users' problems. *Developers should talk to users regularly to better understand their needs.*
- The Product Owner presented a too-granular Product Backlog item instead of seeking a discussion with the Developers about the why and what during refinement. The Developers give the Product Owner what they asked for, not what the user really needs.
- The Product Backlog item was missing acceptance criteria altogether, or existing acceptance criteria missed the problem.

Remedy: No matter the reason, the Scrum Team has to address this critical issue during a Retrospective. This lack of communication and alignment impedes the team's ability to create the best possible value during a Sprint. It is a costly form of misunderstanding.

As a temporary quick fix for the problem, I suggest changing how the Scrum Team creates Product Backlog items: Why not let the Developers write the entries to the Product Backlog? This way, the Product Owner can focus on pitching their ideas to

the other Scrum Team members, explaining their value propositions based on qualitative and quantitative data. It has the additional benefit that a Product Backlog item is now a "team item" and is no longer associated with the Product Owner.

Alternatively, create goal-oriented Product Backlog items; for example, "Build something that achieves [a specified outcome]."

9. Dogmatism Prevents Experimenting

Observation: The Scrum Team applies the Definition of Done also strictly to prototypes and experiments to ensure uniform quality standards.

See Chapter 15, anti-pattern 14, for a detailed description.

10. All Increments Must Contribute to Accomplishing the Product Goal

Observation: The Scrum Team delivers only work items that directly support the Product Goal. All other work is deferred to a later stage.

Background: This is another form of dogmatism with unintended consequences. While the Scrum Guide states that the Scrum Team is focused on one Product Goal at a time,[9] it does not imply that this one goal shall dominate all of the team's work.

Focusing on one Product Goal at a time is beneficial:

- It saves the Scrum Team from context switching and prevents the resulting tax on its productivity.
- It accelerates value delivery and speeds up learning.
- It supports a Scrum Team's work as a cohesive unit.

However, there is likely other work the Scrum Team needs to address. Typical tasks of that category are, for example:

- Fixing bugs
- Running maintenance on the technology stack
- Building prototypes

9. Ken Schwaber and Jeff Sutherland, "Scrum Team," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#scrum-team.

Moreover, suppose your Product Goal does not have a clearcut outcome that signals the Product Goal is done. (An example of such a clearcut outcome would be moving all payment processing to a new provider.)

In this case, the closer a team moves to deliver the current Product Goal, the more urgent the need becomes to start preparing for the subsequent Product Goal. The Product Backlog should not run empty but should smoothly transition from the old to the new Product Goal.

Remedy: As defined in the Scrum Values,[10] *focus* is mission-critical for a Scrum Team. However, in the case of the Product Goal, it does not translate into exclusivity.

If your team needs to quit their current dogmatic approach, I suggest running a Retrospective focused on creating a shared understanding. The Scrum Team should identify the optimal way of allocating their capacity to work on the current Product Goal and to prepare for the next Product Goal, while maintaining the quality of the technology stack.

11. Immediate User Value Is Required in All Increments.

Observation: The Scrum Team focuses solely on smaller work items that provide instant value to users.

Background: Concentrating on short-term value creation has its merits:

- There is instant gratification in delivering something useful, even if it is small.
- Stakeholders will like the team's performance, as every delivery improves their lives.
- The team will generally build trust.

However, focusing on delivering immediate value to stakeholders bears significant risks, too; for example:

- It may also lead to picking low-hanging fruit while ignoring the need to solve complicated and more valuable issues. Those issues typically require more effort and often need to be spread over several Sprints without providing an immediate benefit for customers.

10. Ken Schwaber and Jeff Sutherland, "Scrum Values," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#scrum-values.

- Moreover, the team may not know how to solve the issue; there is a risk of failure.
- And in the long run, the Scrum Team will underperform, as it does not seek to maximize its potential.

Remedy: Focusing on a single-Sprint delivery horizon diminishes the Scrum Team's ability to solve critical customer problems. If your team practices this approach, consider organizing a workshop with the Product Owner that allows the team to revisit the Product Goal or team's product roadmap to understand better the risks and rewards of solving challenging customer problems. The team may come up with a new prioritization of objectives, reflected in the content and order of the Product Backlog.

INCREMENT ANTI-PATTERNS OF THE STAKEHOLDERS OR THE ORGANIZATION

12. Roadmaps Secrets

Observation: The portfolio planning, the release plan, or the product roadmap[11] are not visible to everybody involved in creating Increments.

Background: "If you don't know where you are going, any road will get you there."[12]

The big picture is crucial for any Scrum Team's success. It needs to be available to everybody, as product success is a team sport, dependent on every team member's and stakeholder's ideas, insights, and skills. Excluding team members from this information risks jeopardizing value creation, as they may fail to connect the dots. The same applies, for example, to restricting access to essential data related to the Scrum Team's work, such as the revenue generated by the team's work, utilization of the product or service, customer information, and so on.

11. Release plans and product roadmaps are nothing more than educated guesses about future value creation at a particular moment in time. As they do for the Product Backlog, which represents the near-term subset of the big picture, stakeholders and Scrum Teams must regularly inspect and adapt these release plans and product roadmaps to avoid decoupling from market trends, customer needs, and organizational goals.
12. Lewis Carroll Quotes, BrainyQuote.com, https://www.brainyquote.com/quotes/lewis_carroll_165865.

Remedy: As a Scrum Team, invest appropriately in creating a shared understanding of the big picture, of the why and what. There are several practices that help to meet the challenge; for example:

- Create Product Goals collaboratively. (Toolbox Tip: "Creating Product Goals as a Collaborative Practice" in Appendix B details an exercise for the whole Scrum Team to create, inspect, and adapt a Product Goal.)
- Include the Developers in product discovery activities, such as user interviews and running experiments.
- Turn Product Backlog management and refinement into a collaborative exercise; Developers make excellent Product Backlog item authors.
- Make critical data and information on your product's usage available to everyone; a team wiki works well for that purpose.
- Consider creating information radiators for stakeholders.

The more the contributors are familiar with this background information, the easier the process of maximizing the value of the Scrum Team's work will become.

13. We Know Exactly What the Increment Needs to Be

Observation: The organization believes that rebuilding a legacy application does not require agility but can be planned up front completely. Therefore, the management asks the Scrum Team to revert to a waterfall style of working. After all, it is the same set of functionality.

Background: This approach is a classic rookie mistake, as it deprives customers and the organization of discovering hidden value during the rebuilding process. Also, it is doubtful that the legacy application already represents the highest possible value.

Generally, some issues to watch out for when rebuilding legacy applications are as follows:

- **Missed opportunity for innovation:** Rebuilding a legacy application feature by feature doesn't allow for new ideas, processes, or technologies.
- **Limited optimization:** By focusing on existing features, there's a missed chance to optimize the application by removing redundant or outdated features.

- **Inefficiency:** Rebuilding an application may consume a considerable amount of time, effort, and resources, which the team could invest in enhancing and modernizing the application.
- **Ignored user needs:** Replicating the application may focus on something other than current user needs, leading to a mismatch between what the application offers and what users expect.
- **Inadequate performance:** By merely recreating existing features, the new application may not achieve significant performance improvements, which the team could achieve by rethinking the architecture and design.

Remedy: Rebuilding a legacy application offers an outstanding opportunity not only to replace the technical foundation but also to

- Analyze existing processes,
- Improve usability, and
- Generally make the application simpler to use while unearthing hidden value.

Never let such a splendid opportunity to improve value creation go to waste, because someone believes that your Scrum Team just needs to rebuild the old application.

Food for Thought

Quality and deadlines: In an organization where it is common to sell nonexistent features with delivery deadlines, is there a way to reconcile the Scrum Teams' need for high quality and the business's need to make money? Or will the organization's long-term sustainability always fall prey to the short-term need for closing deals, as the teams must cut corners to deliver, ignoring the Definition of Done, thus accruing technical debt and undone work?

Conclusion

If I had to condense Scrum's purpose to one thing, I would pick delivering valuable Increments every single Sprint. No one really cares whether your Scrum Team is

excellent at product discovery, including stakeholders in decision-making or those collaborating in the spirit of the Scrum Guide.

However, all customers care about whether your team contributes meaningfully to solving their problems. Your stakeholders are interested in that capability, too, compounding this achievement with the expectation that you do so within the given constraints while contributing to your organization's bottom line.

Delivering valuable Increments is the make-or-break of your Scrum Team. Consequently, treat them with attention, dedication, and professionalism.

This page intentionally left blank

13 Product Goal Anti-Patterns

Introduction

Product Goals act as beacons for Scrum Teams and provide transparency on the future state of a product to team members and stakeholders.

As a long-term objective, Product Goals help to connect the dots from product strategy—the big picture—to the Product Backlog, to an individual Sprint and its tactical objective, the Sprint Goal. The Product Goal thus helps the Scrum Team to focus and collaborate, as it reminds everyone on the team of their purpose and why they work together; see Figure 13.1.

Practically, the Product Goal is reflected in the composition and ordering of the Product Backlog. Generally, the rule "add an item to the Product Backlog only if it helps meet the product goal" applies.[1]

1. Roman Pichler, "Product Goals in Scrum," March 9, 2021, https://www.romanpichler.com/blog/product-goals-in-scrum.

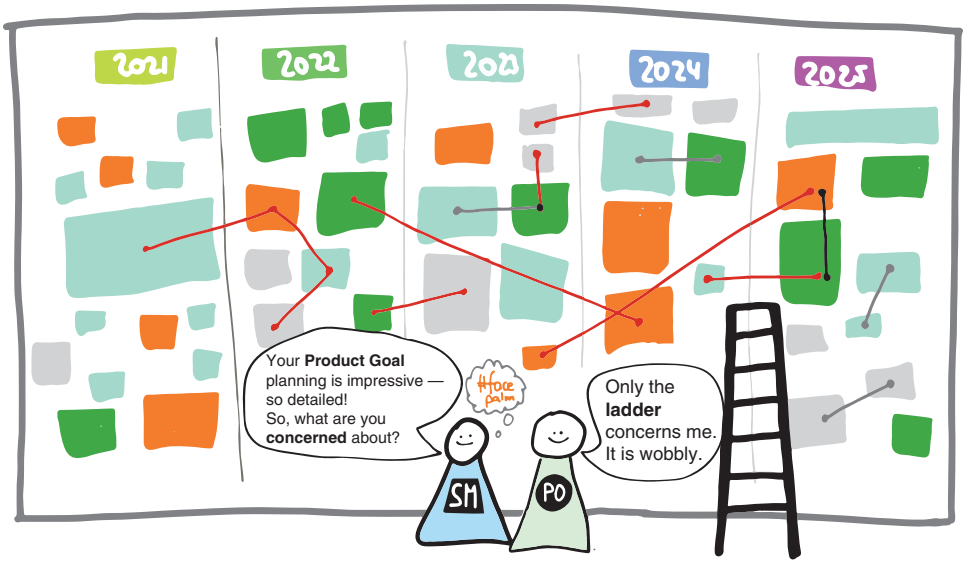

Figure 13.1 Although creating and communicating the Product Goal is the Product Owner's responsibility, collaborating with the Scrum Team helps foster alignment on the why and what. Also, it helps to mitigate the risk that the Product Owner falls in love with their solution instead of the customers' problem.

As a commitment to providing transparency, the Product Goal also contributes to inspecting the Sprint's outcome, inspecting the Increment, and adapting the Product Backlog for the upcoming Sprint, creating alignment between Scrum Team members and stakeholders. The Product Goal helps to answer two critical questions at the Sprint Review:

1. Are we still on track to accomplish the Product Goal?
2. How did the previous Sprint contribute to our team's mission?

The purpose of the Sprint Review is to help the Scrum Team and its stakeholders to answer these questions and to adapt the Product Backlog accordingly.

THE PURPOSE OF THE PRODUCT GOAL ACCORDING TO THE SCRUM GUIDE

The Scrum Guide uses the Product Goal in several sections—from discussing artifacts and commitments to tracking progress to teamwork—to stress its importance

as a unifying element. Indeed, the Product Goal informs the Scrum Team's work from start to finish. This omnipresence is also why successful Scrum Teams ensure a meaningful Product Goal, as these beacons reduce the probability of the Scrum Team running in the wrong direction.

While Scrum anti-patterns at the event level are mainly tactical in nature, Product Goal anti-patterns harbor a higher risk, as they influence the Product Backlog. Choosing an ineffective or misguided long-term objective[2] burdens a Scrum Team with a significant investment risk. Consequently, while a Scrum Team may still be successful overall, even if their Daily Scrum is inadequate, failing to achieve the Product Goal means the team falls short of delivering the value that customers need. For that reason, great Scrum Teams invest significantly in collaboratively creating, inspecting, and adapting their Product Goals.

Product Goal Anti-Patterns

Product Goal Anti-Patterns of the Product Owner

1. There Is No Product Goal

Observation: The Product Owner does not create a Product Goal in the first place. All the Scrum Team's work is short-term and opportunistic.

Background: Without a Product Goal, the Scrum Team will fail to connect the dots from the organization's product strategy to the tactics of the Sprint Goal. As a result, the Scrum Team's ability to deliver value will be impaired. Instead, they will lack focus, lose team cohesion, and be unable to self-manage toward a goal because they have no goal. As the Scrum Guide mentions, the Scrum Team "is a cohesive unit of professionals focused on one objective at a time, the Product Goal."[3]

Remedy: Address the issue during a Sprint Retrospective and help the Product Owner understand how important the creation of a Product Goal is for the long-term success of the product and the Scrum Team.

2. Ken Schwaber and Jeff Sutherland, "Product Backlog," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#product-backlog.
3. Ken Schwaber and Jeff Sutherland, "Scrum Team," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#scrum-team.

Toolbox Tip

"Creating Product Goals as a Collaborative Practice" in Appendix B details an exercise for the whole Scrum Team to create, inspect, and adapt a Product Goal. It uses Ralph Jocham's free Product Goal Canvas.

2. Pursuing Multiple Product Goals

Observation: The Scrum Team elects to work on multiple Product Goals simultaneously.

Background: This approach defies a critical element of using Product Goals successfully: "They must fulfill (or abandon) one objective before taking on the next."[4] The Product Goal provides the Scrum Team with a focus. It reminds everyone on the team why they work together and of its purpose.

Dividing the Scrum Team's attention among several Product Goals will create queues, and disrupt the flow, causing delays and higher costs.[5] The division will reduce the available team capacity for each of the multiple Product Goals. When the team works on Product Goal A, work on all other Product Goals will come to a halt. Consequently, the Product Backlog items related to those other Product Goals will form a queue, "waiting in line" for the Scrum Team to turn its attention to them. While stuck in that queue, those Product Backlog items won't be done and cannot be shipped, further delaying value creation for customers.[6]

Remedy: Scrum Teams focus on the most valuable Product Goal at all times. If your team has to work on multiple Product Goals simultaneously, first analyze the origin of that requirement:

1. If the push comes from the Product Owner, the individual may not know the costs associated with that practice. In this case, educate the Product Owner on

4. Schwaber and Sutherland, "Scrum Team."
5. If you believe that multitasking is an option, please watch this video of Henrik Kniberg. He visualizes the cost and consequences of moving from single-piece flow to simultaneously working on multiple work items: "Multiple WIP vs One Piece Flow Example," Malonus, November 25, 2019, YouTube video, 7:04, https://www.youtube.com/watch?v=Yqi9Gwt-OEA.
6. An excellent book on flow theory in product development is Donald G. Reinertson's *The Principles of Product Development Flow: Second Generation Lean Product Development* (Celeritas Publishing, 2009).

the benefits of faster delivery by focusing the Scrum Team's work on the most valuable Product Goal. (See footnotes 5 and 6.)

2. Alternatively, your organization is responsible for pursuing multiple Product Goals, as your stakeholders regard the Scrum Team as an internal agency to build their requirements. Moreover, they may not respect the role of the Product Owner as the responsible individual for defining and communicating the Product Goal. In this case, the impediment is not merely pursuing multiple Product Goals but a misconception of Scrum basics. Here, educating stakeholders about the benefits of a Scrum Team focusing on one Product Goal may start a discussion about improving the application of Scrum in general.

3. The Product Owner Singlehandedly Creates Product Goals

Observation: The Product Owner creates Product Goals without collaborating with the other Scrum Team members.

Background: While creating Product Goals is part of the Product Owner's responsibilities, it does not mean that the Product Owner needs to handle it separately from the team. On the contrary, to fully benefit from the Scrum Team's expertise, everyone on the team should understand the big (product) picture: Where are they heading, and why do they believe this path is valuable?

Remedy: The Product Owner should include everyone in creating the Product Goal. In a complex environment, there are no experts. Close collaboration with the other Scrum Team members is the best way to ensure that a Product Owner does not fall in love with their own solution, believing they know best what creates value. Instead, including multiple viewpoints from other team members helps to improve the Product Goal by not missing important aspects. Consequently, a collaborative approach to creating Product Goals allows Product Owners to focus on the customers' problems and the organization's sustainability.

4. The Product Owner Forces the Impossible upon the Scrum Team

Observation: For personal reasons, the Product Owner pursues a Product Goal that exceeds the Scrum Team's capabilities, be it skills or resources, and probably even what is possible within the team's technological constraints. As a result, no one believes it is feasible, yet no one on the Scrum Team objects or voices concerns.

Background: There are many reasons why a Product Owner may pursue an ambitious Product Goal; for example:

- They believe in providing "stretch goals" to keep the Scrum Team at peak performance, as stakeholders or the organization's culture regard this as a superior leadership or management style.
- They are eager for a promotion.
- They want to switch jobs and need recommendations in the form of delivering a sophisticated feature. (This happened to one of my Scrum Teams at a prominent startup.)

The challenge is that organizations that disempower Developers also lack psychological safety. Moreover, in such environments, teams rarely live up to Scrum Values. Consequently, people have incentives to follow the path of least resistance rather than speak out when they have no influence on the course of events.

There are several likely results of such a scenario:

- The product will be late and of a lower quality than expected—if it ships at all.
- Personal risk mitigation to avoid the blame game will be on everybody's mind, not how to solve the customers' problems.
- There will be little to no commitment to the "common cause" of the Scrum Team.
- The situation will attract mercenaries who are mainly interested in collecting paychecks and do not care about collaborating on solving customers' problems.

Remedy: Inclusion and psychological safety—which Scrum Values help to achieve—are paramount for innovation and value creation in complex environments. If the Product Owner insists on pushing the Scrum Team beyond its capabilities, address the issue in a Retrospective. If your Product Owner disagrees that their perspective is counterproductive, consider seeking help from an outside Scrum Master or coach. If that approach fails, take it to the management level and ask for help. (Please note that in some company cultures, often sales-driven cultures, the organization considers unachievable objectives to be a suitable means of maximizing the return on investment in a Scrum Team. Under these circumstances, I doubt that applying Scrum is a sustainable choice.)

5. The Product Goal Is Not Transparent

Observation: The Product Owner does not make the Product Goal transparent. Also, the Product Owner does not communicate the goal frequently to increase awareness of the big (product) picture among team members, members of collaborating Scrum Teams, the leadership and management levels, and stakeholders.

Background: Helping every stakeholder and team member fully understand the Product Goal is a major objective for the Product Owner to ensure alignment among all factions contributing to value creation. Failing to do so unnecessarily increases the risk of slowing progress toward the Product Goal. Furthermore, without a transparent Product Goal, inspecting Increments at the Sprint Review will be incomplete.

Remedy: It is an excellent practice to overcommunicate the Product Goal at all opportunities, for example, at the beginning of the Sprint Review. Never expect a stakeholder to be fully aware of what the Scrum Team is attempting to accomplish; they also have their day jobs.

Another good communication tactic is for the Product Owner to summarize the outcome of Scrum events such as Sprint Planning and Sprint Review for stakeholders—even if that is not part of the Scrum Guide's guidance. Doing so provides an excellent opportunity to repeat the team's current understanding of how to accomplish the Product Goal time and again. Additional helpful techniques are, for example:

- An internal team newsletter for stakeholders
- A public dashboard or information radiator
- A team wiki or blog

PRODUCT GOAL ANTI-PATTERNS OF THE SCRUM TEAM

6. Product Goal Exclusionism

Observation: The Scrum Team accepts only Product Goal-related work into the Product Backlog.

Background: While the Product Backlog reflects the Product Goal in its composition and ordering, it can contain items not strictly related to it. Other work items are acceptable as long as they do not interfere with achieving the Product Goal of the Scrum Team. Examples include maintenance work such as refactoring, bug-fixing,

and updating the technology stack, or prototyping. Moreover, other unexpected incidents, such as responding to a security breach, might require the Scrum Team's attention.

Also, when a Scrum Team is closing in on accomplishing the current Product Goal, it prepares for the follow-up Product Goal in parallel to avoid frantic upfront planning on the first day of the first Sprint of the successor Product Goal. Depending on the kind of product the Scrum Team builds, this phase of overlapping focus may take one to three Sprints.

Remedy: A Scrum Team should avoid allocating 100 percent of its capacity to the current Product Goal, ignoring everything else. Instead, reserve regularly some capacity for unexpected or necessary work outside of pursuing the current Product Goal.

7. The Scrum Team Does Not Abandon Obsolete Product Goals

Observation: The Scrum Team sticks with the original Product Goal, although it has become obsolete.

Background: Conceptually, there are several similarities between the Sprint and Product Goals. For example, Scrum Teams do not follow the original Sprint plan when the context has changed and the Sprint goal no longer provides value. As the Product Owner can cancel a Sprint, so can the Scrum Team abandon a Product Goal: "The Product Goal is the long-term objective for the Scrum Team. They must fulfill (or abandon) one objective before taking on the next."[7]

Think of the Product Goal as the overarching hypothesis of the Scrum Team on what is worth building. Given the tactical nature of the Sprint Goals, you can consider them as experiments of the Scrum Team on its path to validate or falsify its hypothesis, expressed in the Product Goal. Consequently, inspecting the outcome of a Sprint allows the Scrum Team to review progress toward the Product Goal at a tactical level. Moreover, based on that learning, the Scrum Team may conclude that the current Product Goal is no longer worth pursuing, as a different Product Goal may offer a better level of value creation.

In such cases, the decision to abandon a Product Goal becomes much easier when it is a collective effort, despite the Product Owner creating and communicating the Product Goal as a part of their Product Backlog management. The prerequisite for

7. Schwaber and Sutherland, "Product Backlog."

this favorable approach is that the Scrum Team fully understands the big (product) picture.

Remedy: Identifying an obsolete Product Goal should not be the sole responsibility of the Product Owner. It is less of a challenge when the Scrum Team does so collectively, for example, during the Sprint Review.

Generally, it pays to support the Product Owner as a team, sharing their burden of figuring out what is worth building. Moreover, this collaborative approach starts way before formulating the Product Goal. For example, we talk about

- Product discovery, as in interviewing customers,
- Building prototypes, and
- Accepting the responsibility of Product Backlog management as a Scrum Team task.

If your Scrum Team hasn't yet reached that level of collaboration, run a Retrospective on how the team can change its current approach. Your Product Owner will be grateful when the team relieves them of the duty of Product Backlog item author; they have so much more on their plate.

8. Reintroducing Waterfall Planning through the Backdoor

Observation: The Product Goal reintroduces waterfall planning through the backdoor as the Scrum Team creates the complete Product Backlog in advance.

Background: While the Scrum Guide states that the "Product Goal is in the Product Backlog," it does not imply that the complete Product Backlog needs to be created in advance: "The rest of the Product Backlog emerges to define 'what' will fulfill the Product Goal."[8]

Remedy: Accomplishing your Product Goal will likely comprise several Sprints. Once a Scrum Team decides to work on a Product Goal, it will create a Product Backlog by focusing on the first one or two Sprints to better understand the problem and solution space. Think of an individual Sprint as an experiment to prove the validity of the overall hypothesis, the Product Goal.

8. Ken Schwaber and Jeff Sutherland, "Scrum Planning," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#product-backlog.

If the team plans too far ahead, it risks creating waste by anticipating work items that may never become necessary to accomplishing the Product Goal. Consequently, the Scrum Team must avoid creating a comprehensive Product Backlog but instead take time to learn what is worth building, mitigating risk by working on small Increments. It is build-measure-learn over and over again.

PRODUCT GOAL ANTI-PATTERNS OF THE ORGANIZATION

9. The Product Goal Predetermines the How

Observation: The Product Goal not only defines what the Scrum Team should accomplish but also imposes a way of realizing this goal.

Background: The issue here is probably that, without consulting the Scrum Team, an outside stakeholder defines a solution path that becomes part of the Product Goal. There are many suspects for this kind of behavior, often exercising a level of control:

- Someone from the C-level, such as the CEO or founder
- Someone from the engineering organization, such as a software architect
- Managers
- Customers

This Product Goal anti-pattern may also result from someone inside the Scrum Team; for example, a "lead Developer."

This anti-pattern creates a particular form of disempowerment of the Scrum Team, as "no one else tells [the Developers] how to turn Product Backlog items into Increments of value."[9] It is an arbitrary interference with the work of the Scrum Teams.[10]

---

9. Schwaber and Sutherland, "Sprint Planning."
10. In addition, this principle anti-pattern—an outsider defining the how on behalf of the Scrum Team—may overlap with particular team topologies in some organizations, distinguishing between
    - Product teams that solve customer problems, and
    - Platform teams that provide infrastructure solutions to product teams.

    Here, the solution path has to become part of the regular constraints of the Scrum Teams, as there is a specific way teams in this organization build products.

Remedy: Maximizing a Scrum Team's impact requires it to be in control over technical solutions and thus accountable for implementing Increments. Externalizing this accountability hence undermines Scrum's core principle of self-management.

To alleviate the issue, I recommend reaching out to the engineering organization, advocating to include the Scrum Teams in the technical decision process. Ideally, this process is embedded with the Scrum Teams themselves in the long run.

> ***Practical Tales:*** *I worked with a large automotive supplier with about 200 people, organized in Scrum Teams, working on a new platform: a marketplace for data in the cloud. Three years into the endeavor, the project was riddled with technical debt, throwing bugs lefts and right. Despite the best efforts, the project could barely move forward.*
>
> *In a project-wide Retrospective, the first public one ever, we identified the central issue of this disaster: sales, product management, and software architects decided, without consulting any Scrum Teams, what needed to be built, when, and how. More often than not, they had already sold the new features to clients. Then, they informed the Scrum Teams about a usually arbitrary delivery date, and Developers started working frantically, cutting corners along the way to save time and meet the deadline. They rarely did.*

10. The Stakeholders Dictate the Product Goal

Observation: The Product Owner does not create the Product Goal, but the stakeholders task the Scrum Team with delivering the Product Goal; see Figure 13.2.

Background: According to the Scrum Guide, creating the Product Goal is part of the Product Owner's Product Backlog management tasks: "The Product Owner is also accountable for effective Product Backlog management, which includes Developing and explicitly communicating the Product Goal."[11] Disempowering the Product Owner by predefining the Product Goal impedes the Scrum Team. It diminishes its potential for self-management and autonomously solving customer problems within the given constraints while contributing to the organization's sustainability.

11. Schwaber and Sutherland, "Product Backlog."

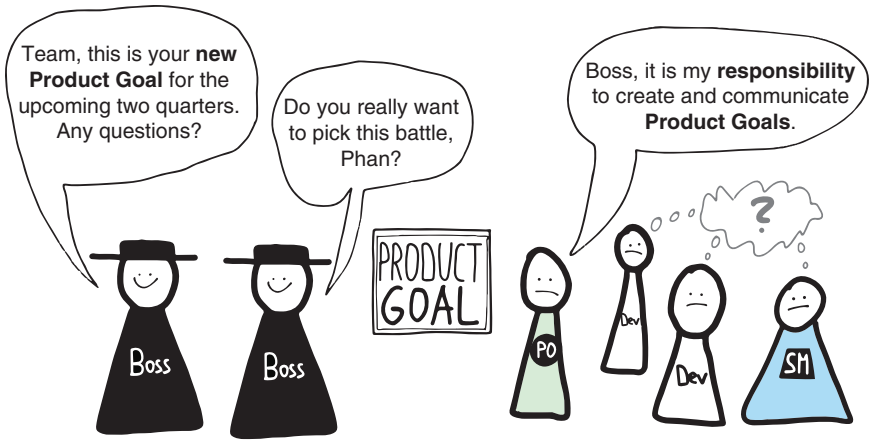

Figure 13.2 The Product Owner's prerogative is to create and communicate the Product Goal; the Scrum Team is not an internal agency for stakeholders.

Remedy: Scrum delivers a superior return on investment compared to a preplanned project approach only when an organization puts Scrum's principles, namely self-management, at its core. Autonomous teams, working closely with customers and directly on the product, are best positioned to explore how to solve the customers' problems by creating hypotheses and testing them by delivering small Increments. In a complex environment, this approach is superior to approaches in which stakeholders, believing that they know best, dictate what a Scrum Team will build based on their assumptions.

Organizations that dictate the Product Goal do not understand or practice agile product development; they merely regard their Scrum Team as tactical delivery entities. When you see this, I suggest reaching out to your stakeholders and pointing them to the opportunity to increase value creation by running an experiment: let the Product Owner decide on one Product Goal and the Scrum Team choose how to best solve the underlying customer problem.

Otherwise, if the organization considers itself to be agile and still treats Scrum Teams as internal delivery agencies, try to understand why this is still the case before reaching out for support from the management level. (Often, in such situations, it will be impossible for a single Scrum Team to progress on the issue. Instead, solving this issue requires the concerted action of all teams.)

Food for Thought

Multiple Product Goals: Scrum Teams work on one Product Goal at a time until it is accomplished or abandoned. Is it advisable to consider multiple Product Goals in advance to smooth the transition from one Product Goal to the next? If so, how many Product Goals do Product Owners plan ahead while avoiding replicating the former yearly planning cycle?

Long-term objectives: According to the Sprint Guide, the Product Goal is a "long-term objective for the Scrum Team." How does "long-term" translate into a planning horizon of the Scrum Team, given that the team is working on complex, adaptive problems—two, four, six, or even more Sprints? Moreover, how do we avoid rejuvenating the classic planning cycle from the waterfall era if we stretch our Product Goal too far?

Conclusion

"If you don't know where you are going, any road will get you there."[12] Lewis Carroll's quote perfectly describes the importance of the Product Goal as an enabler for a Scrum Team's success. If you can fix only one Scrum anti-pattern, my choice would always be the Product Goal, as it influences everything else.

Toolbox

- **"Creating Product Goals as a Collaborative Practice"** suggests using Ralph Jocham's Product Goal Canvas to create a shared understanding among the Scrum Team members about the why, what, and how of the team's journey.

12. Lewis Carroll quotes, BrainyQuotes.com, https://www.brainyquote.com/quotes/lewis_carroll_165865.

This page intentionally left blank

14 Sprint Goal Anti-Patterns

Introduction

The Sprint Goal is a Scrum Team's single objective for the Sprint,[1] delivering the most valuable result from the customers' and the organization's perspective. It informs the composition of the Sprint Backlog and becomes a part of it, thus acting as a beacon that guides the Developers during the Sprint. Moreover, it is instrumental to creating the Sprint Plan, having a successful Daily Scrum, and collaborating and supporting each other as a Scrum Team in general; see Figure 14.1.

Also, the Sprint Goal helps the Scrum Team to identify whether their work was successful: at the end of the Sprint, they ask themselves, "Did we accomplish the goal?" In that respect, it separates a few weeks of working on "stuff" from experiencing the satisfaction and joy of being a successful Scrum Team, delivering value to customers and the organization.

The Sprint Goal thus supports a Scrum Team—and its organization—to move from an industrial paradigm-driven output orientation, the proverbial feature factory,[2] to

1. Ken Schwaber and Jeff Sutherland, "Sprint Backlog," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#sprint-backlog.
2. John Cutler, "12 Signs You're Working in a Feature Factory," @johncutlefish's blog, November 17, 2016, https://cutle.fish/blog/12-signs-youre-working-in-a-feature-factory.

an outcome-based approach to solving customers' most valuable problem every Sprint.

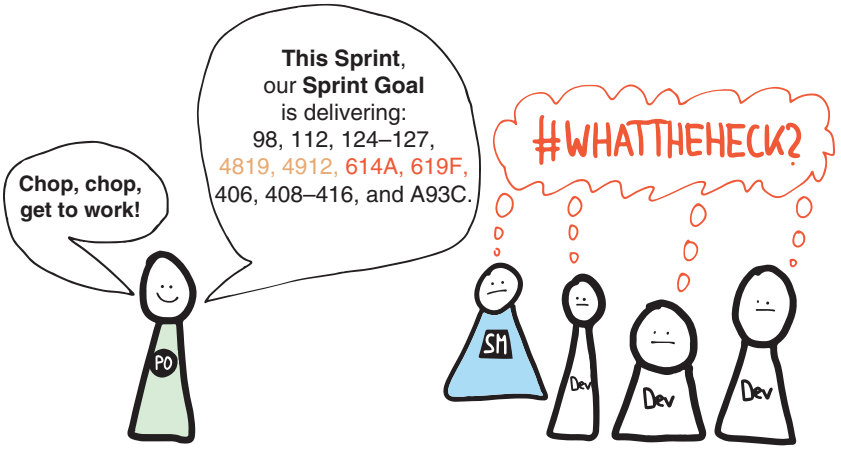

Figure 14.1 The Sprint Goal is not defined by the Product Owner but collectively by the Scrum Team. Also, it is not a list of work items but serves as the flag around which the Scrum Team rallies.

This change of perspective has a far-reaching consequence: every Sprint, the Scrum Team strives to accomplish the Sprint Goal, which is different from maximizing the output in the form of work hours or the number of work items.

How to Create Sprint Goals

Practically speaking, the Sprint Goal originates from the Sprint Planning, which aims to align the Developers and the Product Owner on what to build next, delivering the highest possible value to customers during the upcoming Sprint.

First, the Product Owner points to the team's Product Goal and introduces the business objective of the new Sprint. Based on that information, the Scrum Team then collaboratively creates the Sprint Goal, considering, for example:

- Who is available during the Sprint?
- Are new team members joining the Scrum Team, or are current team members leaving?
- What is the required quality level as defined in the Definition of Done?

- Is the Scrum Team familiar with the necessary technology, or are they lacking skills?
- Does the Scrum Team have all the tools they need?
- Does the Scrum Team have any particular governance requirements?
- Does the Scrum Team have to support daily operations, such as keeping the product up and running? If so, how much capacity will that require?

The Developers commit to the Sprint Goal. Please note that the Developers are not committing to a specific workload, for example, based on the composition of the Sprint Backlog at the end of the Sprint Planning. Scrum is outcome based, not output oriented.

Next, the Developers forecast the work required to achieve the Sprint Goal by choosing items from the Product Backlog and transferring them to the Sprint Backlog. If the Developers identify as yet unknown work necessary to achieve the Sprint Goal, they add it to the Sprint Backlog. They then create an initial plan for the first two or three days of the Sprint to help them get started toward their Sprint Goal. Because they will be learning once they get to work, they don't create a detailed plan for the whole Sprint at this time, as that would be wasteful.

Sprint Goal Anti-Patterns

Unfortunately, many Scrum Teams regard the Sprint Goal as optional, as it often poses a challenge, starting with its creation.[3]

Moreover, Sprint Goals require a supportive organizational environment. Orienting a team's work toward valuable outcomes is particularly challenging in organizations that consider Scrum Teams more of a delivery entity, the feature factory mentioned earlier, than an autonomous team tasked with solving customer problems within the existing constraints.

Both factors feed into creating some of the most prevalent anti-patterns.

3. Learn more about Sprint Goals with Maarten Dalmijn's book *Driving Value with Sprint Goals: Humble Plans, Exceptional Results* (Pearson, 2023).

SPRINT GOAL ANTI-PATTERNS OF THE SCRUM TEAM

1. The Scrum Team Has No Sprint Goal

Observation: The Product Owner proposes a list of Product Backlog items for the Sprint that is simply an arbitrary assortment of tasks without cohesion. Consequently, the Scrum Team cannot create a Sprint Goal.

Background: If choosing a set of Product Backlog items first and trying to create a Sprint Goal based on those items feels natural, you are not really getting value from Scrum as a product development framework. You don't need Scrum if you know precisely what work items create the most value during the Sprint. If you know what work you want to do and simply want to manage it more closely, a flow-based system like Kanban may better help you achieve your aims.

Remedy: Experiment with organizing your Sprint Planning as follows (see also the earlier section "How to Create Sprint Goals"), creating a Sprint Goal in the process:

- The Product Owner points to the Product Goal and the business objective of the upcoming Sprint.
- The Scrum Team collectively creates a Sprint Goal.
- The Developers commit to the Sprint Goal.
- The Developers pick the necessary work to accomplish the Sprint Goal and make an initial plan.

2. The Sprint Goal Is Imposed upon the Scrum Team

Observation: The Scrum Team does not collectively decide on the Sprint Goal, but an individual imposes their idea upon the team.

Background: In these cases, that individual is often a head strong Product Owner or the lead Developer. Other team members, probably due to a lack of psychological safety, do not oppose this act even when they know they will likely not accomplish the Sprint Goal.

The anti-pattern may also indicate that the Scrum Team is unable to live up to Scrum Values. Some members may have given up changing things that do not work and are no longer interested in continuously improving as a team. Probably, the team is not a Scrum Team in the first place but a group where the majority strives to collect a paycheck at the end of the month.

Alternatively, the Scrum Team may not (yet) be empowered to choose a Sprint Goal. In traditionally structured organizations, convincing stakeholders, whose budgets fund a Scrum Team, that the team decides on what they plan to work on is challenging. However, a Scrum Team is not an agency working on a retainer.

Remedy: This issue deserves a dedicated Retrospective to explore the root cause of the anti-pattern, as it undermines the Scrum Team's capability to act as a team. Without becoming an effective team, the Scrum Team will be unable to achieve the teamwork's benefits such as increased productivity and morale. Scrum thrives on including every team member, giving them a voice in the decision process.

3. The Sprint Goal Is Overly Ambitious

Observation: The Scrum Team decides to work toward an overly ambitious Sprint Goal, which leads to an inflated Sprint Backlog. Consequently, the Scrum Team fails to deliver a done product Increment at the end of the Sprint; see Figure 14.2.

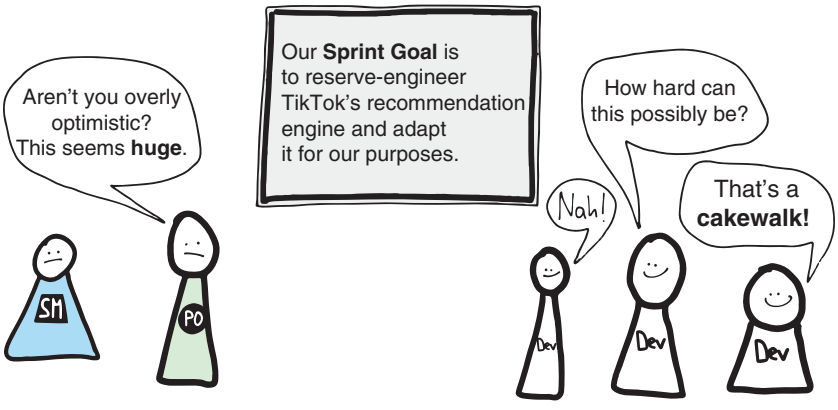

Figure 14.2 Scrum Teams, from time to time, overestimate their capacity and underestimate the effort, thus failing to deliver. However, if not meeting Sprint Goals is the team's norm, it is advisable to identify the reason. To build stakeholder trust, regularly delivering Sprint Goals is critical for a Scrum Team.

Background: It is common for a new Scrum Team to overestimate its capacity and choose unachievable Sprint Goals. However, as the team gains more insight into the problem and solution space of their customer issues, these outliers should subside. An experienced Scrum Team can align its capabilities with its ambition to deliver the best possible value to customers and the organization.

Remedy: Should a Scrum Team continue choosing too-large Sprint Goals, they need to analyze the underlying problems. For example:

- It could result from a Product Backlog refinement process unsuited to meet the required quality levels.
- The team may underestimate the resulting workload from meeting the Definition of Done.
- The team's tech stack might be riddled with technical debt that regularly interferes with accomplishing work items.
- The team may not be good at estimating its capacity.

Use a Retrospective or two to run postmortems, figure out what caused the misjudgment, adjust working agreements accordingly, or invest in more training.

4. The Scrum Team Fails to Achieve the Sprint Goal on a Regular Basis

Observation: The Scrum Team misses the Sprint Goal more often than delivering it.

Background: Scrum Teams are not paid to practice Scrum; they are paid to solve customers' problems within the given constraints, allowing their organization to build a sustainable business.

Given Scrum's delivery focus and tactical nature, agreeing on a Sprint Goal and then achieving it is the most critical contribution of the Scrum Team. While there may be many situations where the team cannot accomplish the Sprint Goal because of, for example, technical issues, skill deficits, and the complexity and uncertainty of life itself, this failure should be the exception, not the rule. If a Scrum Team finds it acceptable to fail to deliver a value product Increment at the end of their Sprint, they have completely missed the purpose of Scrum. From an organizational perspective, a Scrum Team exchanges a forecast for delivering value for inclusion in decision-making, autonomy, and self-organization. Creating a low-grade timeboxed Kanban and calling it Scrum will not honor this deal.

Remedy: Take this issue to a Retrospective and work on identifying the underlying problems that lead to failure to achieve the Sprint Goal regularly. Some causes may be as follows:

- The Scrum Team lacks skills, technical expertise, or tools to accomplish Sprint Goals and ignores the Scrum value of commitment and the need to improve as a team continuously.

- Stakeholders push the Scrum Team too far permanently, and no one cares or dares to point at the issue.
- Stakeholders and line managers consider not having a Sprint Goal a nonissue as long as the Scrum Team delivers a sufficient number of Product Backlog items every Sprint, justifying its existence.
- The team applies Scrum in a situation where acute operational issues overburden every Sprint and leave little room for focused teamwork.
- The Product Backlog is of low quality.
- The technology stack is riddled with undone work and technical debt.

5. The Scrum Team Changes Sprint Goals in the Middle of a Sprint

Observation: The Scrum Team decides mid-Sprint to change the Sprint Goal.

Background: If nothing helps, read the manual: "During the Sprint, no changes are made that would endanger the Sprint Goal."[4]

While it is acceptable to cancel a Sprint if the Sprint Goal becomes obsolete, and while Scrum Teams should expect to refine the scope of the Sprint as they learn more, they should not change the Sprint Goal mid-Sprint, or their Sprint Goal is meaningless.

Remedy: Not changing the Sprint Goal mid-Sprint may appear dogmatic, but it is a valuable practice for guiding the Scrum Team: it stresses the importance of collectively refining the Product Backlog in time to create a shared understanding among the team members on the why, the what, and the how of the next steps. If the Scrum Team members understand the most likely work for the upcoming two to four Sprints, their need to change a Sprint Goal mid-Sprint will diminish.

6. The Scrum Team Does Not Respect Sprint Boundaries

Observation: The Scrum Team habitually takes on too many tasks and moves unfinished work directly to the next Sprint, as they would when using Kanban.

Background: Sometimes a Scrum Team will need to move unfinished Product Backlog items from one Sprint to the next, so long as the Sprint Goal is achieved. When the Sprint Goal is not achieved and a significant percentage of the Sprint

4. Ken Schwaber and Jeff Sutherland, "The Sprint," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#the-sprint.

Backlog moves from one Sprint to the next as a regular matter of course, their Sprints have ceased to be meaningful. In this case, the Scrum Team may be using a kind of timeboxed Kanban and calling it Scrum.

Remedy: This may be the right moment for the Scrum Team members to ask themselves whether moving to Kanban is a viable alternative. After all, they are not paid to practice Scrum but to solve their customers' problems within the given constraints while contributing to the organization's sustainability. However, if the team still considers Scrum its choice, I recommend putting more energy into Product Backlog refinement and creating meaningful Sprint Goals.

7. The Scrum Team Cannot Accommodate Work That Isn't Related to a Sprint Goal

Observation: The Scrum Team does not consider working on any work outside of what is necessary to accomplish the Sprint Goal.

Background: Collectively agreeing on a Sprint Goal does not mean that the Scrum Team has to exclusively focus on achieving that goal. Alongside achieving the Sprint Goal, a Scrum Team often has to still support customers and the organization. The trick to becoming a successful Scrum Team is balancing these two aspects and knowing when to favor one over the other.

For example, no stakeholder will understand when a severe bug remains unfixed because the Scrum Team is dedicated to accomplishing the Sprint Goal. If the payment function of your e-commerce site is no longer working, you will fix this immediately—no matter what the Sprint Goal says. Scrum is not about following the initial Sprint plan to the letter but about adapting to lessons learned during the Sprint. Building trust among stakeholders is a critical success factor for a Scrum Team, and it may take weeks or months to achieve. However, destroying that trust will take only seconds. Ignoring stakeholder needs outside the current Sprint Goal has proven effective for that purpose.

Remedy: Embrace the necessary flexibility as a Scrum Team when designing the Sprint Goal and consider 10 to 20 percent slack time excluded from the team's capacity consideration during Sprint Planning. If the Sprint progresses smoothly

and the team does not need this reserve, you will undoubtedly find ways to allocate the reserve meaningfully.

8. The Sprint Goal Is Confidential

Observation: Instead of communicating it to stakeholders, the Scrum Team keeps the Sprint Goal a secret until the Sprint Review.

Background: Transparency is critical to enable the empirical process of Scrum. Without transparency, an inspection of progress and artifacts isn't possible, reducing the effectiveness of any subsequent adaptation—all of which is foundational for Scrum's risk-mitigation approach.

Remedy: Reducing the level of transparency by not communicating the Sprint Goal reintroduces complexity and uncertainty to the empirical process, rendering the previous work of the Scrum Team less valuable. Moreover, it will likely negatively impact the team's trust level among stakeholders: Who likes to work with a black box? The solution is simple: always communicate the Sprint Goal to everyone openly.

SPRINT GOAL ANTI-PATTERNS INDUCED BY THE ORGANIZATION

9. Sprint Success Defined by Output, Not Outcome

Observation: The organization values the delivery of a predefined workload more than it values achieving the Sprint Goal.

Background: This Scrum-negating anti-pattern is often caused by incentive structures that reward individuals over teams or by outdated industrial-age management approaches that focus on keeping utilization rates of workers (in this case, Developers) high. When this happens, Sprint Backlogs are likely to comprise work items that deliver the required output with the highest probability.

Remedy: One wonders why such organizations try Scrum in the first place. In a complex environment where we do not know precisely how to solve our customers' problems, where we learn as we move forward, maximizing output is the inferior

approach. Moreover, the incentivizing of output will breed mercenaries who trade time for money without much interest or engagement in what they do. Furthermore, mercenaries have little interest in self-management, one of Scrum's first principles; it is just an additional, unpaid effort comparable to being told what to do.

If you want your Scrum Team to live up to its potential and contribute to improving your customers' lives and the organization's sustainability, you need to overcome this industrial-era output orientation. You need to convince stakeholders and the leadership team that an outcome-oriented approach creates more value in the long run. Doing so means embracing agile transformation and organizational change—perhaps more than you want to, but there is no shortcut!

10. The Scrum Team Lacks Focus

Observation: The organization requires the Scrum Team to address many disparate issues simultaneously, impeding the Scrum Team from creating a Sprint Goal.

Background: If the organization considers the Scrum Team to be an internal delivery agency that will be tasked with whatever the stakeholders need, then Scrum in and of itself is not a good choice. Scrum is about solving complex adaptive problems with self-managing, autonomous teams while mitigating the development risk. Scrum is good at scoring match points; however, Scrum is mediocre at best when outsiders define in detail what the team has to deliver.

Remedy: There are two ways to approach this situation:

1. Try to convince your stakeholders that the Scrum Team can at least allocate half of its capacity to solving previously identified customer problems. (Of course, this requires expertise in product discovery and Product Backlog management on the part of the Scrum Team.) The remaining capacity can then be allocated to working incoherent tasks. The goal of this approach would be to prove to stakeholders over time that the Scrum Team can deliver better results and more value working this way than by doing only what it is ordered to do by the stakeholders.
2. If the first approach is not working for whatever reason, consider using Kanban and save yourself a lot of the planning overhead Scrum employs to mitigate risk.

Practical Tales: *I was working with a Scrum Team that had to allocate about 25 percent of its capacity to keeping the application operational. Technically, it was in bad shape: unstable and prone to throwing bugs. Also, many jobs had to be done manually that could have been automated with better foresight. Anyway, the Scrum Team came up with a good way of self-organizing the inevitable workload:*

- *They agreed that every Developer principally had to allocate 2 hours per day to maintenance.*
- *They also agreed that everyone could freely trade those 2 hours as long as someone covered for them.*

The result was that we quickly established a "marketplace" that aligned demand and supply. It turned out that not everyone hated these small maintenance tasks and sometimes volunteered to handle them. Volunteering to do maintenance tasks then freed other Developers so they could focus on tasks from the Sprint Backlog—it was a win-win situation.

Of course, the Scrum Team considered this 25 percent tax during its Sprint Planning accordingly.

SPRINT GOAL ANTI-PATTERNS BY THE DEVELOPERS

11. Cherry-Picking Product Backlog Items Unrelated to the Sprint Goal

Observation: The Developers choose Product Backlog items totally unrelated to the Sprint Goal for the Sprint Backlog. Consequently, the Sprint Backlog resembles a random assortment of work items based on preference, not the Sprint Goal.

Background: There are several reasons why this effect may happen; for example:

- There is usually no Sprint Goal as a starting point for the selection process.
- The Sprint Goal was not really a goal but just a to-do list (e.g., "For this Sprint, our goal is to deliver Product Backlog items 134, 546, 986, and bug 4711").
- The Developers may need to work on critical technical issues that have been neglected and cannot be ignored any longer.

- They are curious to learn something that might be useful for the future; however, the Scrum Team does not or cannot allocate time to research and development.
- They may also disagree with the product's or service's direction as communicated by the Product Owner.

If none of these scenarios applies, the question is whether the "Scrum Team" is really a team and not just a group of individuals, as their way of collaborating seems to be dysfunctional at many levels.

Remedy: Take it to the team and address the topic during a Retrospective. Identify the root cause of what happened, for example, by using Liberating Structures. Also, consider running an anonymous survey in advance to gather data and shed more light on the topic. (It is helpful to publish the survey results before you start the Retrospective.)

12. No Visualization of Progress toward the Sprint Goal

Observation: The Developers cannot answer whether the Scrum Team will likely accomplish the Sprint Goal at the Daily Scrum.

Background: The Daily Scrum serves one purpose, as it answers a simple question: Is the Scrum Team on track to meet the Sprint Goal? If the Developers do not understand where they are concerning progress toward the Sprint Goal, it becomes less likely that they will achieve it. Successful Sprints result from increasing Developers' confidence over time. They do not materialize miraculously on the last day of the Sprint.

Remedy: Visualizing the progress toward the Sprint Goal is useful. Several options exist for accomplishing this Sprint progress visualization, from Sprint boards in various shapes to burndown charts. (Removing the Developers' task of maintaining a mandatory burndown chart from the Scrum Guide a few years ago did not imply their obsolescence.) If the Developers struggle to visualize the progress toward the Sprint Goal, consider running a workshop on Sprint boards.[5]

5. Stefan Wolpers, "Create Whiteboards for Transparency and Collaboration," June 6, 2018, https://age-of-product.com/create-whiteboards.

13. Gold-Plating

Observation: The Developers increase the scope of the Sprint beyond the Sprint Goal by adding unnecessary work to the Sprint Backlog.

See Chapter 3, anti-pattern 5, for a detailed description.

SPRINT GOAL ANTI-PATTERNS BY THE PRODUCT OWNER

14. What Are We Fighting For?

Observation: The Product Owner cannot align the business objective of the upcoming Sprint with the Product Goal and overall product vision, or there is none in the first place. This failure likely results in an arbitrary and incoherent assortment of tasks. Consequently, the Scrum Team fails to create a meaningful Sprint Goal.

Background: A solid business objective answers the "What are we fighting for?" question, enabling the Scrum Team to create a coherent Sprint Goal, which acts as a beacon that guides the Developers during the Sprint.

Moreover, the business objective is focused and measurable, as the Sprint Goal and the Developers' forecast go hand in hand. For example, a business objective could task the Scrum Team with reducing the churn rate during checkout by 20 percent. Then, when the team has more information on why and when the churn is happening, the team can successfully tackle the business objective.

Conversely, if you know precisely what creates the most value during the upcoming Sprint, you no longer need to use Scrum.

Remedy: This is either a personal problem of the Product Owner or a structural problem beyond Scrum, a sibling of anti-pattern 10, The Scrum Team Lacks Focus (see earlier in chapter): In the former case, helping the Product Owner to better understand goal-setting and intensifying the collaboration on product discovery and Product Backlog management within the Scrum Team may provide a path to solving the issue. In the latter case, reaching out to the stakeholders and trying to employ Scrum to the full benefit of the organization could be an approach. However, in both cases, it is critical to figure out the root cause of the issue.

15. Sprint Cancellations without Consultation

Observation: The Product Owner deems the Sprint Goal obsolete and cancels the Sprint without consulting the Scrum Team.

See Chapter 2, anti-pattern 24, for a detailed description.

16. No Sprint Cancellation

Observation: The Product Owner does not cancel a Sprint whose Sprint Goal can no longer be achieved or has become obsolete.

See Chapter 2, anti-pattern 25, for a detailed description.

FOOD FOR THOUGHT

Changing Sprint Goals mid-Sprint: While we do not endanger the Sprint Goal during the Sprint, renegotiating its scope is acceptable to address issues that have arisen during the work of the Sprint. However, when do we cross the threshold that separates renegotiating the scope from changing the Sprint Goal in the middle of the Sprint—when the Scrum Team will no longer deliver or change 20, 30, or 40 percent of the original work? Moreover, does this discussion matter?

No Sprint Goal, yet Scrum: Can we practice Scrum even though we "fail" to create official Sprint Goals due to the nature of our work (think hardware or training a large language model)? Typically, we cannot deliver within a one-month Sprint anyway. Is this still Scrum, and should we care about answering the question when we enjoy working this way?

CONCLUSION

For a good reason, the Sprint Goal is not optional in Scrum. The spotlight provides transparency to the Sprint Backlog, the flag that allows the team to rally, and the one thing that provides focus and cohesion.

Is the Sprint Goal easy to accomplish as a practice? Unfortunately, it is not. On the contrary, it is a challenge at both the team and the organizational level. However, no Scrum Team has ever been able to reap the benefits of the framework to the fullest extent without making the Sprint Goal a cornerstone of its efforts. Therefore, it is worth investing in a Scrum Team's capability to fully embrace the Sprint Goal's underlying idea.

15 Definition of Done Anti-Patterns

Introduction

The Definition of Done (sometimes shortened to DoD) is one of the three commitments of the Scrum Guide. It describes an Increment's required quality standard to be considered "done" and is a prerequisite for inspecting any Increment.

The Definition of Done serves a crucial function, as it provides the Scrum Team and its organization with a common understanding of the completeness of the Increment. The Definition of Done thus enables stakeholders to close the feedback loop the Scrum Team requires for its empirical process, as it allows inspection of a done Increment.

The idea of the Definition of Done is simple: "done" reflects releasable; it means that the Scrum Team can deliver the Increment into the hands of its customers with acceptable quality and completeness to allow the customers to use the product so that the team can understand whether the product Increment was valuable; see Figure 15.1.

Figure 15.1 The Definition of Done does not comprise quality gates or sign-offs from stakeholders. Instead, the Scrum Team decides when an Increment is done.

A done Increment

- Provides incremental value to customers and improves the utility of a product or service compared to the previous Increment,
- Is safe to ship from a team and organizational perspective,
- Meets the market's quality expectations, and
- Is a crucial element of the Scrum Team's self-managed way of value creation, as Scrum does not provide any approval gates.

Achieving business agility requires organizations to learn faster than the competition and be able to turn that learning into marketable products and services ahead of their competition. To do so, the Scrum Team needs to be able to create product Increments that are usable by customers. The Definition of Done helps Scrum Teams to create product Increments that embody an appropriate level of technical quality to ensure that customers receive a usable product increment.

Applying a Definition of Done does not entail delivering the best possible quality at all times. However, the Scrum Team must provide such a level of quality that the Definition of Done meets the specifications and expectations of all internal and external stakeholders.

The Scrum Guide[1] frequently mentions the Definition of Done to underline the importance of having a widely known and accepted quality standard for a successful Scrum Team and organization. While an adequate Definition of Done does not guarantee a Scrum Team's success, its absence certainly accelerates its failure.

Creating a Successful Definition of Done

A successful Definition of Done follows four basic principles:

1. The Definition of Done is a living document, regularly inspected and adapted by the Scrum Team as it becomes more familiar with the product's or service's problem and solution spaces. The team's initial Definition of Done reflects the organization's standards and the collective experience of the Scrum Team, which it refines and expands, informed by the experience gained from delivering actual Increments to customers. There are various opportunities for a Scrum Team to collect relevant data, from customer feedback at Sprint Reviews to operational data such as the number of bugs that escaped to production.
2. The Definition of Done incorporates organizational guiding principles on technical or quality standards. As the Definition of Done allows for transparency to help team members and stakeholders inspect the quality level of an Increment, there needs to be a baseline understood by everyone to simplify the process and reduce complexity. However, beyond this smallest common denominator of quality, the team can lift the standard if it promises a return on investment. Examples of guiding organizational principles are adherence to architecture guidelines and coding standards at a technical level. Moreover, there could be requirements regarding governance rules; for example, in highly regulated industries, it could be mandatory to receive legal advice before releasing new functionality.
3. When multiple Scrum Teams work on the same product, they must align on a shared Definition of Done that applies to every Product Backlog item. An individual Scrum Team of the collective can choose to lift its team-internal quality standard above the shared standard. Again, this approach helps to reduce complexity.

1. Ken Schwaber and Jeff Sutherland, The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html.

4. There is always only one Definition of Done for a single product or service. Whenever the Definition of Done is updated, the Scrum Team needs to apply the new standard retroactively to all previously created Increments to avoid reintroducing complexity by allowing different quality standards to exist in parallel. There can be only one Definition of Done at any given time.

In a large organization where multiple Scrum Teams work on a single product, each team's Definition of Done builds upon organizational and cross-team Definitions of Done:

- Layer 1 (optional): The organization's guiding principles on technical or quality standards
- Layer 2 (optional): The shared quality standards of a collective of Scrum Teams working on the same product or service
- Layer 3 (required): The individual Scrum Team's team-internal quality standards

This layering of concerns is represented in Figure 15.2.

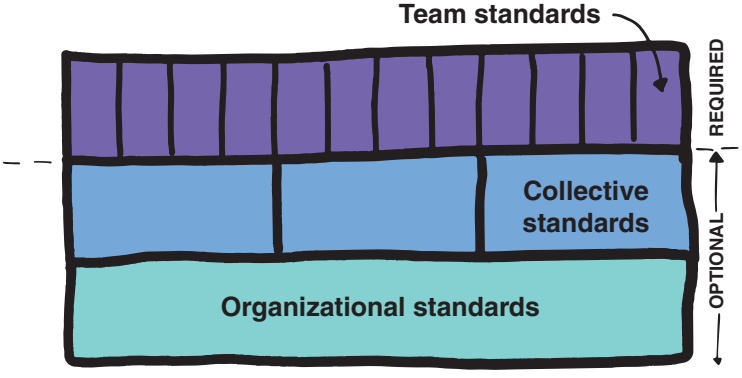

Figure 15.2 The three layers of a Definition of Done.

Toolbox Tip

"How to Create a Definition of Done" in Appendix B details a collaborative exercise of the whole Scrum Team, possibly including some of the stakeholders at a later stage, to create, inspect, and adapt a Definition of Done.

Definition of Done Anti-Patterns

Definition of Done anti-patterns follow recurring themes:

- **Convenient application**: The Scrum Team applies the quality standard when it is convenient but not as a first principle. This sometimes occurs when Scrum Team forecasts are too optimistic and it takes on too much work and an overly ambitious Sprint Goal. When the team members decide to deliver the impossible by cutting back on quality, they violate their Definition of Done because they think it is more convenient to do so rather than reducing their scope and scaling back their goal. This actually creates more work later as they deal with the consequences of poor quality, and it may, in fact, damage both their reputation and that of their product. A Scrum Team either meets the quality standard defined by the Definition of Done or it does not, as the Definition of Done is not simply a "nice thing to do."
- **Subjective interpretation:** Some team members interpret the meaning of *done* generously. If a team member says a Product Backlog item is done, there should be no further need for inspection from a quality perspective. When team members selectively interpret the Definition of Done, the integrity of the product Increment can be sacrificed, along with trust between team members.
- **Bowing to outside pressure:** The Scrum Team ignores its Definition of Done in response to external pressure, such as from management, to deliver more features, faster. While there are situations where emergencies trigger imperative responses, such as when the organization's survival is at stake, caving to an external dictum kills a team's ability to make decisions for itself and damages a team's capability to deliver value in the future. (See also the "Dying in Beauty" controversy in the Food for Thought paragraph at the end of this chapter.)
- **The Definition of Done does not evolve**: A Scrum Team creates a Definition of Done once but never updates it. A Scrum Team needs to continually examine whether it is meeting stakeholder expectations of quality. If it is not, the team needs to improve its Definition of Done. It is also critical that the team updates all previous Increments to match the new Definition of Done. This necessity will take time away from, for example, creating new features. The investment in the longevity of the product's technical foundation will compete for the scarce capacity of the Scrum Team without having an immediate return, which makes updating the old Increments prone to being skipped.

These patterns are further explored in subsequent sections.

DEFINITION OF DONE ANTI-PATTERNS OF THE DEVELOPERS

The Scrum Guide states that Developers are accountable for "instilling quality by adhering to a Definition of Done."[2] With great responsibility also comes great potential for conflict, as some stakeholders may dispute the Developers' accountability, advocating for lower-quality levels to ship faster and less expensively. In these situations, seasoned Developers need to stand their ground, learn to say no without burning bridges, and resist giving in to pressure at the expense of quality. This section contains some DoD-related anti-patterns I have observed.

1. Ignoring the Definition of Done

Observation: The Developers reduce product quality below the level defined in the Definition of Done to meet a deadline.

Background: The Definition of Done serves a critical role in the Scrum framework: it represents the quality standard that signals to everyone what the Developers need to accomplish to create a potentially releasable Increment. By "releasable," we mean that we can deliver the Increment into the hands of our customers and users without any legal, financial, or ethical repercussions.

Watering this quality level down to meet a—probably arbitrary—deadline imposed on the team puts the team's success at risk, as it introduces undone work and technical debt. Both of these consequences of neglecting the quality standard will impede the Scrum Team's ability to be innovative at a later stage.

Remedy: Business agility is only possible with technical excellence. The latter starts with always adhering to the Definition of Done. When the Scrum Team is confronted with unreasonable deadlines, it is of little use to alleviate the problem by deliberately lowering quality. This approach adds complexity and unpredictable risks and costs to the future path of the Scrum Team as undone work compounds and increases the level of technical debt. Moreover, the team needs to understand why the stakeholders impose these deadlines and how it can help the stakeholders understand the resulting negative consequences of creating a low-quality product.

2. Ken Schwaber and Jeff Sutherland, "Developers," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#developers.

Practical Tales: *A prominent startup I supported had a popular sports app with tens of millions of users globally. The startup was venture-capital funded and cash-flow negative. A significant milestone ahead of the next funding round was to introduce a pay-per-view feature to the app. Once the licensing agreement was signed, the iPhone and Android teams got to work. While the iPhone team managed to deliver in time for the planned marketing campaign, the Android team had to concede that it could not provide the feature. During a postmortem, it turned out that the Android folks had not adhered to their Definition of Done for years, accumulating so much undone work and technical debt that shortcuts would no longer work. Instead, they had to rebuild the complete technology stack for six-plus months. Investors, founders, and users were not pleased.*

2. Multiple Versions of the Definition of Done in a Single Product

Observation: An update on the Definition of Done does not apply to all previous Increments. The Scrum Team does not update some older product parts to the new quality standard.

Background: The Definition of Done can work as a transparency-creating commitment only when its respective version applies to all previous Increments. Consequently, the new quality standard must be retroactively applied to all previous work whenever the Definition of Done is upgraded or improved. This application requires an extra effort, which is unpopular with many stakeholders, probably even the Product Owner, as they cannot see the immediate benefit.

Remedy: Ignoring the necessity to update all previous Increments to the new Definition of Done standard contributes to a slowly but steadily declining product quality, thus introducing hidden costs in the form of additional complexity. To fully enjoy the benefits of a robust Definition of Done, there can be only one quality standard throughout the product. If you improve the Definition of Done, upgrade all of your previous work to that new level.

3. The Definition of Done's Sibling

Observation: The Developers create a Definition of Ready as a gate to Sprint Planning.

Background: Generally, there is no need for such a thing as a Definition of Ready. The natural state of Product Backlog items after a meaningful, collaborative refinement is "ready"—the team is reasonably confident that it can turn the item into an Increment during a Sprint. However, in the early stages of the new Scrum Team, it is sometimes helpful to use, for example, a checklist to better understand whether the team successfully refined a Product Backlog item. Once the team is more familiar with the problem and solution space and has created a usable Definition of Done, this checklist loses value.

Sometimes this checklist evolves into something more formal, a Definition of Ready. While it sounds to some like a good thing, it causes a lot of problems to have a strict definition-of-ready gate that causes a Scrum Team to reject Product Backlog items from Sprint Planning that do not conform to its rules.

A Definition of Ready is usually a cover for deeper problems. By blocking consideration of Product Backlog items in Sprint Planning, it is simply delaying the confrontation of these issues:

- The Product Owner may not be sufficiently available to the Developers to answer questions during the Sprint. A Definition of Ready solution is for Product Backlog items to be described in greater detail, but this simply creates waste. A better solution is to find ways to give the Product Owner more time to spend with the Developers.
- There may be trust issues on the team, perhaps resulting from past failures to reach Sprint Goals. A Definition of Ready solution is to create detailed acceptance criteria for each Product Backlog item, but this increases the time and effort invested in Product Backlog items that may not be valuable to customers. A better solution is to address the basic trust and communication issues between team members.
- Product Backlog refinement may not result in Product Backlog items that are achievable in a Sprint. A Definition of Ready solution may specify sizing limits on each Product Backlog item, but this requires the Scrum Team to forecast all items in the Product Backlog. This can be wasteful because some Product Backlog items are unlikely to ever be developed, and it does not address the root cause of why the Scrum Team cannot produce a done Increment (see prior discussions).

Remedy: Don't adopt a Definition of Ready. A better solution to temptations to introduce a Definition of Ready is to analyze the root cause of the problem, starting with a team Retrospective.

> ***Practical Tales:*** *At a former client of mine, a large automotive supplier, the VP of a product group introduced a definition of ready Jira plug-in in an attempt to improve the overall quality of the bug-ridden product. The plug-in produced visually lovely reports, and the teams were evaluated based on improving their "readiness." Unfortunately, tracking a lagging symptom metric did not help with the commonly known root cause, as the Scrum Team was now incentivized to improve this readiness metric. Instead, improving collaboration between stakeholders and teams would have been more beneficial to reduce the bug load and improve the overall product quality.*

DEFINITION OF DONE ANTI-PATTERNS OF THE SCRUM TEAM

In this section, I address ignoring or not meeting the required quality standard. This attitude may result either from a deliberate decision of the whole Scrum Team, or there is a collective, often unspoken agreement to be cavalier about the issue.

4. A Scrum Team Has No Definition of Done

Observation: The Scrum Team does not define its standards for release quality. Release decisions are arbitrary and ad hoc.

Background: Scrum Teams who lack a Definition of Done may see their capability to deliver valuable Increments to customers slowly but steadily decline. Typical manifestations of the lack of a Definition of Done are as follows:

- The Scrum Team finds the Product Backlog refinement process increasingly challenging. The lack of a shared quality level makes understanding the required effort and complexity of individual Product Backlog items difficult.
- The lack of a shared quality level also affects Sprint Planning, reducing the Scrum Team's ability to identify meaningful Sprint Goals.
- Forecasting by the Developers becomes even less reliable because, without a Definition of Done, they do not know what work needs to be accomplished.

To compensate for this effect, they may increasingly only commit to smaller Sprint Goals to create buffer time to deal with inevitable problems.

- The organization's journey to business agility becomes more difficult. For example, it is likely to incur more technical debt, leading to deterioration of the underlying technology stack as issues of undone work compound.[3] A generally declining product quality often results in a higher likelihood of the Scrum Team releasing bugs to production.
- More bugs in the production environment reduce stakeholder trust. In return, the stakeholders may feel obliged to lead the Scrum Team more closely, impeding its opportunity to self-manage.
- To limit the escape of bugs into production systems, the Scrum Team or the organization may attempt to implement extensive testing. However, this costly effort would address the symptoms of the problem—a lack of quality—but not the cause, in this case, a lack of applying a shared quality standard to all work.
- The lack of product quality ultimately brings the visible progress of the Scrum Team to a halt.
- Customers lose confidence in the product and reorient themselves toward competitors.
- The Scrum Team members might lose trust and confidence in their collective work, with the result that their ability to work together is impaired.
- In the long run, the Scrum Team members with an agile mindset may feel the need to look for new employment opportunities.

Remedy: The Scrum Team creates their Definition of Done. Even a simple one is better than none so long as the Scrum Team continually seeks to improve it based on feedback. I recommend to start small, observe the outcome, and then iterate to incorporate the learnings. Check out the Toolbox Tip "How to Create a Definition of Done" at the end of this chapter if you struggle to kick off the creating process.

3. Technical debt is the long-term consequence of suboptimal decisions or shortcuts during software development. These shortcuts often prioritize short-term gains over long-term code quality and maintainability. Over time, technical debt accumulates and negatively impacts a Scrum Team's effectiveness, as additional efforts are required to fix or refactor the codebase.

Practical Tales: *I encountered one exception from the previously suggested rule that a predefined and documented process of creating and maintaining a Definition of Done is essential for sustainable team success. I worked with one exceptional Scrum Team that did not have a formal Definition of Done. Instead, the team*

1. *Accomplished all of its work based on pair or mob programming.*
2. *Meticulously tracked all technical issues that might become technical debt later, even predefining when a particular topic would need an inspection.*
3. *Worked bug-free; the only issues were exotic edge cases.*
4. *Was also really good at forecasting.*

The idea of "quality" was so ingrained in the team's conduct that no one considered it necessary to formalize the approach by defining a Definition of Done. By making quality the first principle of their way of working and embedding it in their culture, the Scrum Team members created a kind of unspoken Definition of Done. It was fun working with them.

5. Lack of Collaboration in Creating the Definition of Done

Observation: Creating the Definition of Done is not a team effort.

Background: The Developers instill quality by adhering to the Definition of Done.[4] The best way to ensure that everyone adheres to it is for them to participate in creating it. They also create a better Definition of Done when they include different perspectives. For example, the Product Owner is often more attuned to customer expectations, and internal stakeholders can help to improve the Definition of Done by including governance or compliance perspectives.

Remedy: The joint creation of the Definition of Done is another example of Scrum's checks and balances. Collaboration reduces the risk that the Definition of Done will miss important perspectives.

6. The Scrum Team Releases Increments That Fail to Meet the Definition of Done

Observation: The Scrum Team decides to release an undone Increment.

4. Schwaber and Sutherland, "Developers."

Background: The Scrum Guide states: "The Definition of Done is a formal description of the state of the Increment when it meets the quality measures required for the product."[5] When a Scrum Team decides to release a product Increment that does not meet their Definition of Done, it shows lack of respect for both itself and its customers and users. The result often damages the reputation of the product and of the Scrum Team.

Remedy: First, help everyone on the team understand why a unified quality standard is critical for the Scrum Team's long-term ability to solve customer problems. Shipping undone work compounds over time, increasing the complexity level the team tries to tame by applying Scrum in the first place. By shipping undone work, the team undermines its efforts to become agile.

Second, there are always exceptions to the rule, but they need to be carefully considered. Imagine an urgent bug-fix in a life-or-death situation. If the potential cost or damage of not releasing is greater than the potential cost or damage of low quality, the Definition of Done might need to be put aside temporarily. There are situations in which deviation is justified, but they should be rare exceptions. If they are not, something else is wrong.

7. The Product Increment Fails to Achieve the Desired Level of Quality

Observation: The Definition of Done is not adequate to meet the expectations of stakeholders and customers. Instead, bugs slip through to production, and trust in the Scrum Team's capabilities deteriorates.

Background: The Definition of Done must match the required quality level for releases: we can deliver the Increment to our customers without legal, moral, ethical, or financial repercussions, such as incurring penalties or becoming subject to liability claims. A done Increment is safe to ship.

There are many reasons why a Scrum Team may be incapable of meeting required quality standards; for example:[6]

- Limited understanding of the problem domain
- Insufficient knowledge of customer requirements

5. Ken Schwaber and Jeff Sutherland, "Increment," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#increment.
6. Learn more about software development practices from Robert C. Martin's *Clean Code: A Handbook of Agile Software Craftsmanship* (Pearson, 2008).

- Subpar problem-solving abilities
- No dedication to technical excellence
- Inadequate expertise in the technology stack
- Weak development capabilities
- Excessive dependence on manual testing or code reviews for detecting quality issues
- Absence of automated testing practices
- Unavailability of automated deployment processes
- Ineffective version control practices, such as allowing untested code to be committed

Remedy: If the Scrum Team's Definition of Done is not effective at ensuring the desired quality level, the Scrum Team needs to take action to improve it. Some suitable questions to start the inspection are, for example:

- Do we rely too much on manual testing or code reviews to catch quality problems?
- Should we invest in automation, such as
  - Automated testing?
  - Automated configuration management?
  - Automated deployment?
- Do our version control practices match quality requirements?

The team may need to dedicate a Sprint Retrospective to resolve this problem.

8. The Scrum Team Produces an Undone Product Increment or One That Fails to Meet the Sprint Goal

Observation: Disregarding the Definition of Done during the Sprint Planning can lead a Scrum Team to take on more work than they can deliver at an acceptable quality level. As a result, their Sprint Goal may be unachievable, or they may decide to release an undone Increment (see anti-pattern 3, The Definition of Done's Sibling, earlier in this chapter).

Background: This is a common affliction of newly formed Scrum Teams; it takes a new team a few Sprints to understand what it can and cannot achieve during a Sprint.

Remedy: When this happens frequently, one or more things may need to be addressed:

- The Scrum Team may need to improve its Product Backlog refinement process so that the Product Backlog items are sized small enough to complete in a single Sprint.
- The Scrum Team may also need to refine its Definition of Done so that it is appropriate and not overly stringent or ambitious.
- The Scrum Team may lack technical skills or is ill-equipped to deal with the challenges the team is facing.

Whatever the reason(s), the Scrum Team needs to investigate the root causes of this effect. A "we aimed high and did not make it—too bad" attitude is not helpful for the Scrum Team's long-term prospects. Consider running a Retrospective to identify the root cause of the issue; the "5 Whys" exercise recommends itself.[7]

9. Playing Safe with the Definition of Done

Observation: The Definition of Done does not balance risk and opportunity. Instead, it is a way for the Scrum Team to play it safe by disguising its concerns as an overly strict Definition of Done; see Figure 15.3.

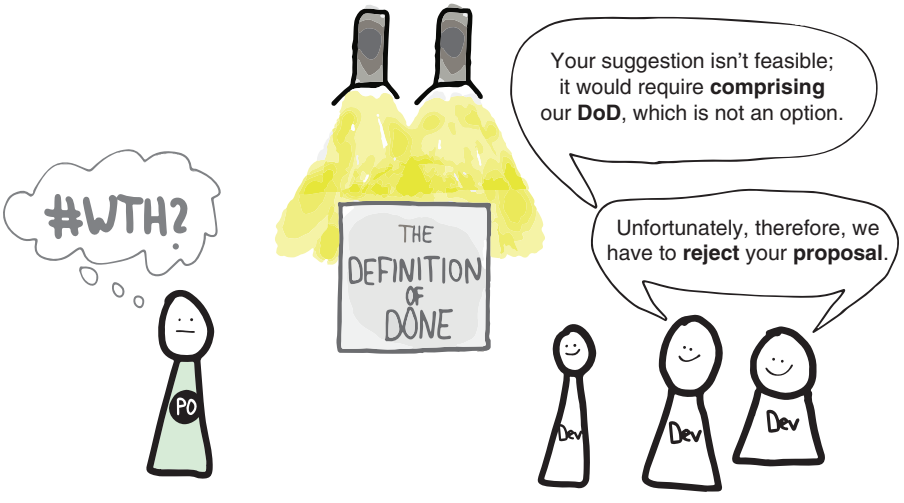
#WTH?
PO
THE DEFINITION OF DONE
Your suggestion isn't feasible; it would require **comprising** our **DoD**, which is not an option.
Unfortunately, therefore, we have to **reject** your **proposal**.
Dev
Dev
Dev

Figure 15.3 Employing a Definition of Done does not imply playing safe by applying the highest quality standard under all circumstances.

7. Peter Horvath, "You Are Using the '5 Whys' Wrong. Here's How to Improve," September 29, 2021, https://medium.com/@petertamashorvath/you-are-using-the-5-whys-wrong-here-s-how-to-improve-d22abf9384d7.

Background: Quality does not always mean perfection; it just needs to be adequate for the job. Consequently, your Definition of Done does not need to reflect the highest possible quality standard. However, if the Scrum Team uses the Definition of Done defensively, it may signal that the team members do not feel entirely supported by the organization in their work, as illustrated in Figure 15.2. Maybe the organization still has a "failure is not an option" attitude, and the team wants to be on the safe side by avoiding the blame game.

Remedy: While we do not strive to fail, failure in and of itself is inevitable in a complex environment. If an organization cannot deal with that fact of life, supporting Scrum Teams will require more work at the organizational level; it is no longer an issue of the Scrum Team.

10. The Definition of Done Is Too Demanding

Observation: The organizational quality guidelines make the Definition of Done too ambitious, considering the needs of the organization and customers, to allow for the regular and frequent delivery of Increments.

Background: A Definition of Done can be too stringent, precise, and detailed. When it is beyond the ability of a Scrum Team to adhere to it, the team may ignore or bend it. This may cause the reputation of the team to decline within the organization and among customers.

Remedy: An unattainable or unreasonably strict Definition of Done is just as bad as having none. The Scrum Team needs a Definition of Done that it can actually follow. If it is too stringent, the team needs to consider inspecting and adapting it to a manageable level.

When some other part of the organization, such as a compliance or governance organization, demands a too-stringent Definition of Done, Scrum Teams need to reach out to them to understand why they require such an unattainable level of quality. Once the stakeholders learn how they influence the team's capability to solve the customers' problems, the Scrum Team may be able to negotiate less demanding requirements.

11. Too Much Ambition, Too Soon

Observation: The Scrum Team's Definition of Done is branching out to adjacent areas too early, thus involving the team unnecessarily in conflicts with stakeholders.

Background: A good team—as it learns more about the problem and solutions space of its product or service—branches out into neighboring areas of the organization, thus improving quality levels over time while speeding up delivery and smoothing the flow of work. Doing so too early, however, often results in the Scrum Team's distraction. If the stakeholders' perception of the Scrum Team's expertise and its ability to deliver Increments differs from the team's own perception, the step will likely be controversial. Stakeholders may regard the team's ambitions as a kind of power grab rather than as a legitimate ambition to deliver more value with higher predictability and frequency.

Remedy: As a Scrum Team, focus on building trust among stakeholders first by regularly providing high-value Increments before aiming to extend your Definition of Done beyond the team's immediate sphere of influence.

12. Not Improving the Definition of Done

Observation: The Scrum Team does not regularly inspect the Definition of Done and discuss whether it needs to improve.

Background: As the saying goes: Standing still is moving backward. As the Scrum Team's understanding of how to solve its customers' problems improves, its Definition of Done needs to evolve in parallel. Otherwise, the Scrum Team would risk diminishing its contribution to the organization's overall path to business agility for two reasons:

1. The Scrum Team will not reach its full potential, delivering less value than possible over time.
2. The Scrum Team may lose its advantage over competitors or fail to catch up, probably falling behind even further.

Remedy: The Scrum Team should agree on a regular schedule for inspecting and adapting its Definition of Done. Also, consider identifying a directly responsible individual to take care of this issue, reminding the Scrum Team in time to inspect the Definition of Done. This comes in handy to detect when quality is slowly but steadily declining, which may not be evident to everyone on the team. Support this approach by tracking appropriate metrics, such as the number of bugs escaping to product environments, or lead time and cycle time.

Definition of Done Anti-Patterns of the Organization

No guidance on how to apply a Definition of Done can be as damaging as providing too much direction. Therefore, we need to have a look at the organization as well.

13. No Common Ground in a Product Group

Observation: Multiple Scrum Teams working on the same product from the same Product Backlog do not share a basic Definition of Done as the smallest common denominator.

Background: This anti-pattern is a variation of the previously addressed Multiple Versions of the Definition of Done in a Single Product anti-pattern, lifting this complexity-increasing problem to even more problematic heights. For example, an individual Scrum Team may have some understanding of where undone work resides or where technical debt has been accumulated. However, when multiple teams contribute to the final product, this level of transparency and understanding between all teams is unlikely to happen. Consequently, the collective will likely add complexity at an accelerated rate compared to a single team, making the product more prone to technical failure.

Remedy: If your Scrum Team is part of a collective of Scrum Teams, a product group, perhaps using LeSS or Nexus to coordinate their work, ensure that all teams in the collective adhere to the commonly agreed-upon overarching Definition of Done. Otherwise, the product group may face even worse challenges than a single Scrum Team having multiple versions of the Definition of Done in its single product.

14. Dogmatism Prevents Experimenting

Observation: The Definition of Done is also strictly applied to prototypes and experiments to ensure uniform quality standards.

Background: This approach is not just costly, it also defies a vital principle of product discovery: you want to experiment and validate or falsify hypotheses as cheaply as possible. In that respect, throwaway prototypes make excellent vehicles for acquiring knowledge. But, of course, this is only the case if you apply common sense and ignore parts of your Definition of Done for this purpose for a limited time.

Remedy: Do not apply your standard Definition of Done strictly to building prototypes. Doing so would negate the idea of validating hypotheses with experiments. Instead, use common sense regarding the level of quality needed depending on the fidelity grade of the prototype: a click dummy does not require adhering to the Definition of Done; a high-fidelity prototype with live data does to a certain extent. However, ensure you refactor everything once you turn the prototype into a real feature. A lot of technical debt is created by ignoring this necessity.

15. No Transparency

Observation: The Definition of Done is not available to everyone in the organization; it lacks transparency.

Background: As a common quality standard, the Definition of Done needs to be open to everyone within the organization. Otherwise, how would stakeholders, for example, be able to inspect the Increments at Sprint Planning?

Remedy: Therefore, it is an excellent idea to overcommunicate the Definition of Done, making accessing it as simple as possible. For example:

- Visualize the Definition of Done (e.g., bring a poster of the Definition of Done to the Sprint Planning and Sprint Review).
- If you invite your stakeholders to the Sprint Review, include a link to the Definition of Done with your invitation.
- Provide leaflets with the current version of the Definition of Done for the Sprint Review attendees.

DEFINITION OF DONE ANTI-PATTERNS OF THE PRODUCT OWNER

Finally, the Product Owner's contribution to the Definition of Done anti-patterns, besides not participating in creating it, is the idea of "accepting" work from the Developers.

16. "Acceptance" by the Product Owner

Observation: The Product Owner uses the Sprint Review to "accept" Product Backlog items that Developers consider done, as they adhere to the team's Definition of Done.

See Chapter 2, anti-pattern 29, for a detailed description.

Food for Thought

Dying in Beauty: Your startup has little remaining runway before going out of business due to the lack of funds. In such situations, should a Scrum Team skip its strict adherence to the Definition of Done in favor of extending the runway in the hope of making another milestone that convinces investors to participate in a new funding round?

Land-grab: Scrum Teams that become more knowledgeable in their problem and solution space tend to enhance their Definition of Done into areas outside their immediate responsibility to improve value creation. How far should they take this idea? Is seeking the confrontation with other stakeholders worth the effort, or is it a mere distraction of the Scrum Team?

Governance: In a regulated industry, can Governance and Scrum Teams coexist peacefully by integrating the and contractual requirements into the Definition of Done? Or will the respective Governance body always constitute an external approval gate unknown to Scrum?

Conclusion

The Definition of Done is an essential stepping stone for the Scrum Team to deliver an Increment of expected quality. It provides a good return on investment from the team's perspective and should guide the Scrum Team toward accomplishing its Product Goals.

Creating and maintaining a Definition of Done is playing a long game that reflects the Scrum Value *commitment*. As a Scrum Team, we want to constantly improve how we work, maximizing the long-term value creation for our customers while contributing to the organization's sustainability. Therefore, the Definition of Done is a critical investment of the team.

Neglecting the Definition of Done, on the other hand, will slowly but steadily erode the team's capability to solve the customers' problems, its reputation, and its contribution to the organization's longevity.

This page intentionally left blank

Appendix A

How to Sabotage Scrum Masters and Product Owners at an Organizational Level

The Exercise

The underlying exercise for the revelations of this chapter is based on a Liberating Structure microstructure called TRIZ.[1]

TRIZ fosters innovation by prompting groups to acknowledge and discard habits hindering progress. This tool encourages daring thought by safely questioning established norms. Asking, "What must we stop doing to make progress on our deepest purpose?" stimulates crucial yet enjoyable discussions. Through creative creative discussion, opportunities for renewal arise as localized, innovative actions replace outdated practices.

The idea of TRIZ is rooted in reframing a familiar situation by changing the perspective. For example, typically, we would approach a challenging situation that bears room for improvement during a Retrospective by asking what does not work to our satisfaction. Then, once we analyzed the issues, we would come up with suggestions for improving the situation.

1. Learn more: Henri Lipmanowicz and Keith McCandless, *The Surprising Powers of Liberating Structures* (Liberating Structures Press, 2013): 187–90.

TRIZ turns this sequence upside-down: First, we think about how to create an unwanted outcome deliberately. By unleashing their inner Darth Vader, participants creatively gain new insights into the structural issues of the problem. In that respect, TRIZ works similarly to, for example, premortems.[2]

I run this exercise—how to sabotage a Scrum Master or Product Owner—in all my Professional Scrum Master and Product Owner training classes for two reasons:

1. I want to introduce TRIZ as an effective exercise with an example from everyday professional life.
2. The results from the exercise allow Scrum Masters and Product Owners to understand possible organizational challenges better.

So, once we accomplish the first part of the exercise—aggregating sabotaging practices—the workgroup then identifies those practices to sabotage Scrum Masters and Product Owners they have already witnessed, followed by identifying possible ways to counter these practices.

Here is the briefing for the participants; typically, they have a 5-minute timebox to come up with suggestions and observations in a workgroup of three to five people.

- *You are a middle manager in the IT organization, and you believe Scrum is a fad and will go away—with a little help from your side.*
- *Your task: As a team, come up with ideas on how to best sabotage the new [Scrum Master or Product Owner] of the first Scrum Team in your organization.*
- *Please note that only legal and culturally accepted practices qualify as a good outcome.*

I aggregated the following suggestions on how to best sabotage Scrum Masters and Product Owners at an organizational level from more than 50 classes.

2. A premortem is a proactive risk management strategy whereby a team imagines a project has failed, then works backward to anticipate and mitigate potential issues that might lead to this negative outcome. Learn more: Shreyas Doshi, "Pre-mortems: How a Stripe Product Manager Predicts & Prevents Problems before Launch," 2020, https://coda.io/@shreyas/pre-mortems-how-a-stripe-product-manager-predicts-prevents-probl.

43 Ways to Sabotage a Scrum Master

You can categorize the results from the aforementioned Scrum Master classes into six groups. See Figure A.1.

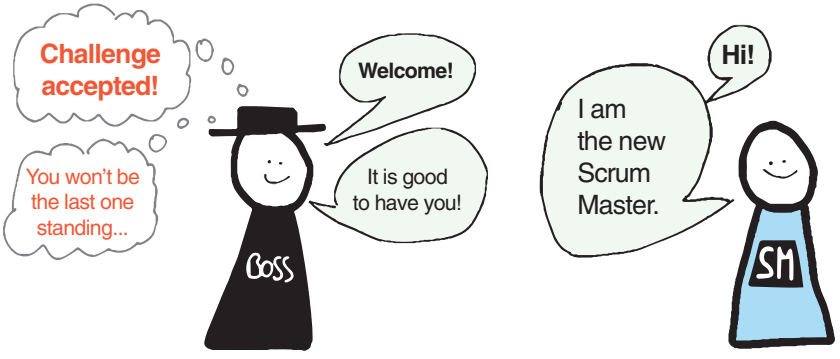

Figure A.1 Not everyone believes that empowered, self-managing Scrum Teams is the way forward.

1. Messing with the Scrum Framework in General

A good way to sabotage a Scrum Master is to disqualify using Scrum as a helpful framework or change Scrum in ways that conflict with the first principles of Scrum:

- Place the blame on Scrum whenever you can, even if it is technically unrelated.
- If Scrum uncovers an obstacle in the organization, blame that on Scrum.
- Share examples of where Scrum failed in other companies.
- Speak disrespectfully of Scrum whenever you can.
- Challenge anything that Scrum Masters say or do.
- Decline offers to learn about Scrum.
- Create an ego-centric incentive system.
- Assign multiple Product Owners to a Scrum Team.
- Place a proxy Product Owner in the Scrum Team and overrule their decisions.

2. Metrics and Reporting

Other useful sabotage practices are to use metrics, objectives and key results (OKRs), and key performance indicators (KPIs) to undermine your Scrum Master

by turning them into a highly paid data-entry clerk with a challenging reporting burden:

- Ask the Scrum Master to prove their value with metrics.
- Create performance KPIs for each team member.
- Ask the Scrum Master to collect all working hours of the team members.
- Ask for individual performance metrics for every Sprint.
- Tie team member performance reviews with their average story points per Sprint using the Bell curve.
- Punish the Developers for not meeting their estimates.

3. Scrum Team Building and Management

If a challenging reporting burden does not help, sabotage your Scrum Master by actively undermining their efforts to turn a group of people into a cross-functional Scrum Team:

- Only add members to the Scrum Team who don't live any Scrum values.
- Recommend the most bullheaded senior developer as the engineering team lead.
- Constantly switch Developers from one project to another, claiming emergencies that require swift action.
- Have Scrum Team members regularly work on multiple different Scrum Teams.
- Add new people to the Scrum Team without consulting the team.
- Slow down the hiring or replacement processes.
- Promote a member within the Scrum Team to act as a proxy manager.

4. Work Organization

If you are already messing with the Scrum Team's team-building process, why not place a few obstacles into the Scrum Team's way of working? Sabotage your Scrum Master by creating unachievable objectives while meddling with the very foundation of Scrum:

- Define unachievable objectives for the Scrum Team.
- Overload the Scrum Team with requests, then complain to others that you do not get results on time.

- Hand over only fixed-price, fixed-time, and fixed-scope projects to the Scrum Team.
- Change requirements during the Sprint.
- Insist on hard deadlines for deliveries of features.
- Ask the Scrum Master to provide a product roadmap with milestones and deadlines.
- Make the Scrum Master responsible for accomplishing milestones and deadlines.
- Outsource part of the product roadmap creation to an off-site team working in a completely different time zone.
- Ask Developers to do work that switches their focus away from the Sprint Goal.
- Assign tasks directly to Scrum Team members.
- Don't allow Scrum Team members to speak directly to customers; act as the single point of customer contact.
- Create unnecessary organizational bottlenecks outside of Scrum, such as approval gates.
- Don't provide adequate equipment and tools to the Scrum Team.

5. Flow of Information

Does your Scrum Master have an insatiable appetite for data, information, and knowledge? Well, keep them out of the loop then. What could be an easier way of sabotaging Scrum:

- Claim that everyone already knows what to do, and therefore there is no need for team member alignment or a Scrum Master.
- Don't share essential or valuable information with the Scrum Team.
- Establish a strict need-to-know basis for sharing information to reinforce organizational silos.

6. Scrum Events and Other Meetings

Finally, ensure that everyone on the Scrum Team understands that your events are more important than theirs:

- Claim that there are too many Scrum events and that they take too much time. Instead, suggest skipping some of them.

- Require that you be present at every Scrum event.
- Exclude the Scrum Master from important meetings outside the Scrum Team's events.
- Pull Scrum Team members into long meetings for which their presence is not necessary, especially during their Scrum Team events.
- Be very understanding of the needs of the Scrum Team; for example, that stakeholders shall participate in the Sprint Review. However, never join any Scrum event yourself.

53 WAYS TO SABOTAGE A PRODUCT OWNER

I categorized the results from the aforementioned Product Owner classes into eight groups. See Figure A.2.

Figure A.2 Not everyone appreciates the Product Owner's authority regarding the product.

1. MESSING WITH THE SCRUM FRAMEWORK

The first category of how to best sabotage a Product Owner is generally about disqualifying Scrum itself as a helpful framework or introducing changes to conflict with the first principles of Scrum:

- Ignore the autonomy of the Product Owner.
- Bypass the Product Owner and talk directly to the Developers.

- Force the same person to be both Product Owner and Scrum Master.
- Impose traditional project management tasks on the Product Owner.
- Enforce traditional or waterfall practices within the Scrum framework; for example, insist on using the Sprint Review to approve Increments.
- Isolate the Scrum Team by applying "old" processes—like approval gates—at the interface between the Scrum Team and the rest of the organization.
- Organize important stakeholder meetings at the same time as Sprint Reviews so that stakeholders cannot attend Sprint Reviews.
- Ensure that critical stakeholders are never available for the Product Owner.
- Impose nonnegotiable hard timelines or milestones.
- Insist that projects be accomplished within a fixed time period, with a fixed scope and a fixed budget.

2. Scrum Team Building and Line Management

If the direct approach does not yield the expected results, why not use the backdoor and help make the Scrum Team as dysfunctional as possible? Here are some suggestions worth considering:

- Don't let Scrum Teams self-manage; instead, micromanage them.
- Assign Developers to multiple projects to decrease their effectiveness, then blame the team for lack of productivity.
- Frequently change the members of Scrum Team so they are constantly in a state of team building. For example, introduce new team members during a Sprint or shuffle Developers between Scrum Teams.
- Assign new members to a Scrum Team who openly despise Scrum.
- Assign ad hoc tasks to team members who are outside the Scrum Team's focus.
- Criticize the Developers. Tell them they are too slow and they don't understand the business.
- Create contradicting individual incentives for Developers to create internal competition within the Scrum Team.
- Use the Daily Scrum to check on progress, and hold the Product Owner accountable for not meeting your targets.
- Deny the Scrum Team the tools and other resources they need.

3. Flow of Information

If you are already messing with the Scrum Team's team-building process, why not place a few obstacles into the Scrum Team's way of working? Sabotage your Product Owner by keeping them in the dark while holding them accountable for not being up to date:

- Don't provide feedback.
- Respond irregularly and with long delay to inquiries of the Product Owner.
- Don't share new market insights or intelligence with the Product Owner, particularly during the Sprint Review. Later, hold the Product Owner accountable for not knowing this information.
- Don't attend the Sprint Review at all.
- Don't accept meeting requests by the Product Owner.
- Restrict the Scrum Team's access to the customers.
- Ensure that only salespeople are allowed to relay feedback from customers.
- Never communicate organizational changes that could affect the Scrum Team's work in time for them to respond to it.

4. Metrics and Reporting

The next category of useful sabotage practices are metrics, OKRs, and KPIs:

- Keep the Product Owner busy with unnecessary reports, documentation, or administrative tasks.
- Enforce proprietary and obscure metrics[3] that are useless to the Scrum Team.

3. While students did not provide specific metrics, some that come to mind are: *Total user signups: Number of Website Visits: Emails Sent to Customers: Average session duration: Number of support tickets submitted:*
   - *Total user signups:* This metric only measures the number of people who signed up for your product, not how many people use or derive value from it.
   - *Number of Website Visits:* The raw number of website visits doesn't tell you anything about user behavior or engagement on your site.
   - *Emails Sent to Customers:* The number of emails sent doesn't provide valuable insights unless tied to engagement metrics such as open rates, clickthrough rates, or conversion rates.
   - *Average session duration:* Longer sessions don't necessarily indicate higher user engagement; they could mean users struggle to find what they need.
   - *Number of support tickets submitted:* A high number of tickets can be a sign of product issues, but it could also be due to inadequate user education or complicated processes.

- Demand evidence of progress toward the Product Goal. At the same time, reject all metrics from the Product Owner and the Scrum Team as unsuitable for your needs.
- Demand detailed task-level plans for all the work the Scrum Team will do and when they will do it.

5. Vision, Product Discovery, and Product Backlog Management

All sabotage is—at least partly—based on deception. Therefore, obfuscate whatever might help the Product Owner to get an understanding of what strategic direction their Scrum Team is supposed to support:

- Constantly change the organization's focus or priorities.
- Create an unclear vision and keep changing it regularly. Better yet, don't have a vision and focus instead only on cost and efficiency.
- Create unclear requirements and refuse to elaborate, claiming they are self-explanatory.
- Overwhelm the Product Owner with a never-ending flow of unclear requirements and requests.
- Ask for requirements specifically outside of the expertise of the Scrum Team, deny them the necessary resources, yet hold the Product Owner accountable for the subsequent failure.
- Overrule the Product Owner's decisions.
- Spread the Product Owner thin by assigning them to multiple products, then criticize them for their resulting lack of focus.
- Add items to the Product Backlog without going through the Product Owner.
- Micromanage the Product Backlog and its refinement; for example, insist on being present but rarely commit to attendance.
- Insist on detailed specifications for each Product Backlog item.

6. Messing with the Sprint, Increment, or Release

Does the Product Owner still lead the Scrum Team toward creating Increments? Time for some rearguard action by introducing new approval stages and security measures:

- Reject Increments as not meeting specifications, and blame the Product Owner.

- Add tasks for the Developers to the Sprint Backlog that are not aligned with the Sprint Goal.
- Keep striving for perfection, never release.
- Impose an external release approval process on the Scrum Team.
- Install a separate governance board to oversee product quality and security issues.
- Define new security protocols that overburden the Scrum Team, claiming legal requirements.

7. Politics

Use your career equity, the network you have been building for years, to your advantage. Maybe it is time to call in some favors:

- Seed doubt about the capabilities of the Product Owner with managers and executives.
- Challenge Scrum's suitability as a framework in your organization's case by pointing at failures in other organizations.
- Play the blame game with the Product Owner for any failure or shortcoming while claiming successes of the Scrum Team as yours.

8. Budget

If everything fails, don't waste time reading the Scrum Guide. Instead, use the budgeting process to your advantage:

- Slash the budget but keep the scope.
- Keep the budget but increase the scope.
- In general, make the budget process longer and more complex by adding useless requirements, for example, regarding the paperwork.

Conclusion

Every manager has ample opportunity to sabotage Scrum Masters and Product Owners while seeming to be simply exhibiting managerial best practices. Providing the benefit of the doubt and assuming positive intent, these tactics may prove successful.

Moreover, not everyone will be excited when the organization succeeds in becoming agile. Self-management and shifting decision-making to those closest to the problems will likely flatten the hierarchy, probably reducing the need for managers, as successful transformation often results from descaling the organization.

Identifying these behaviors early helps Scrum Teams to counteract their effects before they cause substantial damage.

This page intentionally left blank

Appendix B
Toolbox

20 Questions from a New Scrum Master to the Scrum Team

Some time ago, a prospective client asked me to participate in the Product Backlog refinement of a Scrum Team looking for a new Scrum Master. Naturally, I was skeptical in the beginning. I had limited knowledge about the project, a commercial website based on a content management system; the refinement session was time-boxed to 60 minutes; and I hadn't met the team members beyond a very brief hello.

So, I prepared a questionnaire comprising team-related topics. Wanting to learn more about the project and team, I listened to the Scrum Team refining several work items. When appropriate, I asked one of the prepared questions. Surprisingly, the insights turned out to be much more qualified than I expected. I could quickly identify the low-hanging "Scrum fruits" to improve product delivery. Your stakeholders will always judge you by whether "your" Scrum Team can deliver a valuable, potentially releasable, done Product Increment regularly.

The following 20 questions[1] for you—the new Scrum Master—fit into a 60-minute timebox. Start learning how your new Scrum Team is currently delivering the

1. Stefan Wolpers, "Download the 60 Questions for the New Scrum Master Questionnaire," July 5, 2021, https://age-of-product.com/download-60-questions-new-scrum-master-questionnaire.

product and get up to speed, from Product Backlog forensics to metrics to team challenges and technical debt:

1. **How large is your Product Backlog?** I do not believe in Product Backlogs that are larger than the team can handle in three or four Sprints. If the Product Backlog exceeds this threshold, the Product Owner might need some support.
2. **What is the typical age of a Product Backlog item?** A Product Backlog item that has not been touched in three months is obsolete. Furthermore, cluttering the Product Backlog with ideas, reminders, or other low-value items increases the noise, thus probably impeding the value delivery of the Scrum Team. (Admittedly, I am a fan of Product Backlog forensics.)
3. **What is your average lead time from an item being added to the Product Backlog to its delivery?** No one could answer that question in the aforementioned session. It is one of only a few metrics that can provide insight into whether your organization has successfully adopted Scrum.
4. **Does your Product Backlog contain items that none of the current team members refined?** If so, the Product Backlog items in question may no longer be valuable. Consider re-refining or deleting them.
5. **How often are you refining the Product Backlog?** That should be a continuous process. As a new Scrum Master, however, I would also love to have a Product Backlog refinement session with the team once a week.
6. **How many Product Backlog items are you working on in parallel during Product Backlog refinement?** Ideally, a team should be working only on work items it can handle within the next two or three Sprints. Otherwise, the risk of allocating resources to work items that may never make it into a Sprint Backlog becomes too high.
7. **How long does the refinement of a typical Product Backlog item take?** The refinement should take at most one to two Sprints. Otherwise, the Product Owner might need some support properly preparing ideas for the refinement sessions.
8. **How are you creating Product Backlog items?** (Is it a joint team effort with the Product Owner, or is the Product Owner writing the work items, and the Development Team then estimates them?) There is a tendency for a Product Owner to become a "technical writer" of Product Backlog items, which the Developers then estimate; this approach is less effective than turning Product Backlog item creation into a joint effort of the whole team.

9. **Where are you discussing Product Backlog items?** (Only during refinement sessions, Slack, or via comments on tickets, for example?) Every team has its habits, and commenting in Confluence, Jira, Github, or utilizing Slack is an effective means of communication in your organization. This collaboration should be fine if it happens before the Developers select a work item for a Sprint Backlog. However, discussing its essentials afterward is a problem, as the Sprint Goal might be compromised.
10. **Who is writing the acceptance criteria, and in what format?** It should be the Product Owner in collaboration with the Developers, following the Definition of Done, an approach that creates a shared understanding of what the Scrum Team needs to build.
11. **How are you estimating the likely effort of a Product Backlog item?** There are plenty of practices for estimating the size of a work item, if your Scrum Team estimates at all. (Think of #noestimates, or create similarly sized work items instead.) The emphasis should be on creating a shared understanding among all team members of what shall be created. An estimate, the number, is a side effect, not the primary purpose.
12. **Are you estimating in person-hours or story points?** Estimating in person-hours is better than not estimating at all. However, I prefer story points, particularly if the application is burdened with legacy code, technical debt, or undone work. In addition, predictability and stakeholder communications become more manageable as story points have a built-in buffer, covering both effort and complexity.
13. **How are you practicing the estimation process if the team shares different opinions?** Preferably, you should observe the team's estimation process in real life. But in case you have to ask: Is it a typical vote-discuss-revote cycle? Or, when and how do you pick an estimate? (Examples are a 50:50 split [e.g., three times a 3 and three times a 5]—which one do you take? Or a majority split [e.g., two times a 2 and four times a 5]. Or do the estimations cover a range [e.g., from 2 to 8]?) How do you deal with bias—are the less-experienced team members following the lead of the more-experienced team members? Learning about the estimation process is also an excellent way to learn more about the cohesiveness of the team.
14. **What is a typical distribution of work item sizes in your Sprint Backlog?** This question tries to figure out, based on the Sprint Backlog composition, where the productivity sweet spot of the team is. In my experience, Scrum Teams often work more successfully when a Sprint Backlog comprises one or two larger Product Backlog items, some medium-sized stories, and a few small ones.

15. **Are you re-estimating work items at the end of a Sprint?** If so, under what circumstances are you doing so? Scrum Teams should always consider revisiting a Product Backlog item if it turns out to be way off its original estimation. Re-estimates make good data points for Sprint Retrospectives, too. Think of them as micro postmortems.
16. **What was your velocity for the last three Sprints?** A Scrum Team should know its velocity or productivity; how could it improve otherwise? As a team-internal metric, velocity is a reasonably helpful metric.
17. **How many work items are typically not finished within a Sprint, and for what reasons?** If the Scrum Team is overly optimistic and picked more Product Backlog items than it could handle at the beginning of the Sprint, it's not a problem so long as the Developers meet the Sprint Goal. If the Developers, however, are regularly leaving work items on the board and not meeting Sprint Goals, the Scrum Master should be concerned.
18. **Are you changing Product Backlog items once they become part of a Sprint Backlog?** And if so, under what circumstances? For example, reducing the scope of a Product Backlog item if the Developers run into a problem is certainly not great but is still acceptable if the revised work item still delivers value and the Scrum Team meets the Sprint Goal. However, extending a work item after Sprint Planning is permissible only if the Developers agree to the extension and the Sprint Goal will not be compromised.
19. **How would you consider the level of technical debt?** As a new Scrum Master, you want to know everything about the current level of technical debt and dependencies on other Scrum Teams or external suppliers. These issues are directly responsible for your Scrum Team's ability to deliver valuable Increments.
20. **What are the top three challenges the Scrum Team is facing today?** This closing question is, by design, an open-ended question to get some ideas for the next Sprint Retrospective.

20 QUESTIONS FROM A NEW SCRUM MASTER TO THE PRODUCT OWNER

If your Scrum Team fails to deliver value because they are not delivering things that customers value, your Product Owner needs help in identifying and ordering Product Backlog items.

This set of questions[2] helps the new Scrum Master understand how the Product Owner works with the Product Backlog and the rest of the Scrum Team to uncover potential areas for improvement. I modeled these questions after some of the basic characteristics that high-performing teams have in common. They are intended for the first talk between the Product Owner and a new Scrum Master without involving the whole Scrum Team:

1. **What is the product vision, and what is the corresponding go-to-market strategy?** The Product Owner must be able to provide the rest of the Scrum Team with a description of what the product will do for customers and how the customers will learn about and obtain the product.
2. **How do you learn about new ideas and requirements?** Here, the Product Owner should explain the product discovery process: from ideas to hypotheses to experiments, and finally, to validation.
3. **How do you include user research in the product discovery process?** I don't believe a learning organization can manage without continuously running user tests. Therefore, the Product Owner should explain the user research process and how the Scrum Team is involved in product discovery.
4. **How much time do you allocate to user research and understanding your customers' needs?** As a rule of thumb, 50 percent would be excellent. If it's less than 10 percent, there is (a lot of) room for improvement for the Product Owner. Follow up with a question on the Product Owner's workload and how you can support the Product Owner to free up more time for talking to customers.
5. **At what stage do you involve the Scrum Team in product discovery?** The Product Owner should engage the Scrum Team in the product discovery process from the initial discussions of the vision, even if it's only sketchy. The Product Owner may adjust the timing depending on the team's experience, but sooner tends to be better. The sooner the Developers participate in the process, the less the chances are that the Product Owner will pursue solutions that are technically not viable or would not result in a return on investment.
6. **How do you organize the collaboration with stakeholders?** This question should be easy to answer for the Product Owner. There are various ways to establish and improve this communication over time. For example, institute regular meetings with each stakeholder. Or have stakeholders name product ambassadors who act

2. Wolpers, "Download the 60 Questions."

as liaison officers and train them. Arrange workshops with stakeholders or ambassadors. Team up with the user experience people and run, for example, user journey or user story mapping workshops. Or invite stakeholders to Product Backlog refinement sessions to explain the value of a work item.

7. **How do you deal with pet projects[3] that are of little value regarding the Scrum Team's Product Goal?** You want to know that your Product Owner can guard the Product Backlog. Has the individual said no in the past? And if the Scrum Team has worked on pet projects, why is that so?
8. **What is your approach to creating product roadmaps?** Product roadmap[4] planning—as with portfolio planning—is a continuous exercise. The Product Owner should be able to detail the process of the organization.
9. **How large is your Product Backlog?** I do not believe in Product Backlogs that are larger than the team can handle in three, maybe four Sprints.[5] However, if the Product Backlog exceeds this threshold, the Product Owner might need support to prevent this potentially disadvantageous cluttering of the Product Backlog. After all, we are looking for valuable signals in the Product Backlog.
10. **What is the typical age of a work item in the Product Backlog?** A Product Backlog item that has not been touched in three months is obsolete. "But I have been working on it since" is an excuse in my eyes. Understanding the age of each Product Backlog item, who created it, who commented on it (if anyone), who changed it (if anyone), can help you and the Product Owner better understand the way the team and the organization regard or work with the Product Backlog.
11. **What is your average lead time from identifying a potentially valuable idea to adding the corresponding, validated work item to the Product Backlog?** The lead time is one of a few metrics that can provide insight into the effectiveness of the product discovery process.
12. **Does your Product Backlog contain work items none of the current team members refined?** If so, the Scrum Team members should revisit the Product

3. A pet project is an initiative or idea that team members or stakeholders passionately advocate for, often due to a personal interest or belief in its potential, regardless of whether it aligns with the broader organizational goals or the Scrum Team's Product Goal.
4. A product roadmap is a strategic planning tool that visualizes how a product will likely evolve across several product versions, providing context around the product's development. It aids alignment among stakeholders and guides development efforts toward achieving the product vision. Learn more: Roman Pichler, *Agile Product Management with Scrum: Creating Products That Customers Love* (Addison-Wesley, 2010).
5. Learn more about the Product Backlog management process in Chapter 10.

Backlog item to ensure that (a) the work items would still provide value today and (b) there is a shared understanding among all team members regarding the nature of the item.

13. **How often are you refining the Product Backlog?** That should be a continuous exercise. However, it has proven helpful to have at least one regular refinement event per Sprint in which the whole Scrum Team participates.
14. **How many Product Backlog items are you working on in parallel during Product Backlog refinement?** Ideally, a team should be working only on work items it can handle within the next two or three Sprints. Otherwise, the risk of allocating resources to work items that may never make it into a Sprint Backlog becomes too high.
15. **How long does the refinement of a typical work item take?** The refinement should extend only one to two Sprints; context-switching perils and work-in-progress limits also apply to refinement sessions.
16. **How are you creating work items?** Is it a joint team effort, or is the Product Owner writing the user stories and the Developers estimating them? There is a tendency for a Product Owner to become more of a technical writer of work items, which the Developers then estimate. I strongly suggest turning Product Backlog item creation into a joint effort of the whole team to create a shared understanding. You want every item in the Product Backlog to be a collective item of the Scrum Team, not one handed down from the Product Owner to the Developers.
17. **How are you discussing work items?** Only during refinement sessions, Slack, or via comments on tickets, for example? Every team has its habits, and commenting in Confluence, Jira, or Github or utilizing Slack is an effective means of communication in your organization. If this discussion happens before the Developers select a Product Backlog item for a Sprint backlog, it should be fine if the Product Owner agrees, too. Discussing its essentials during the Sprint for the first time might be a problem.
18. **Are you changing Product Backlog items once they become part of a Sprint Backlog?** And if so, under what circumstances? For example, making work items smaller if the Developers run into a problem is certainly not great, but it is acceptable—if the work item in its reduced form still delivers value and the Scrum Team therefore meets the Sprint Goal. However, extending it after the Sprint Planning would be tolerable only if the Developers agree on the extension and the Sprint Goal will not be compromised.

19. **When do you accept work items?** The Scrum Guide does not mention "acceptance" by the Product Owner. However, discussing the state of a work item between the Product Owner and the Developers has proven beneficial when they consider the work item meets the Definition of Done.
20. **Have you ever rejected work items?** If so, you want to learn why that happened. Moreover, you want to know what measures the Scrum Team took to prevent it from happening again.

20 Questions from a New Scrum Master to the Developers

From Scrum Master to Developers, this set of questions addresses the foundations of a Scrum Team's capability to build valuable products: technical excellence and what it takes to achieve this proficiency level. I modeled these questions[6] after basic principles that high-performing teams have in common: from keeping technical debt at bay to sharing knowledge generously to collaboratively creating a Product Backlog.

The following questionnaire has proven helpful in a team discussion: as a guide during an individual interview or as the basis for an anonymous survey. Either way, the 20 questions from the new Scrum Master to Developers get the discussion going:

1. **Who is creating your Definition of Done?** Typically, it is created either by the organization or by the Developers in collaboration with the Product Owner. The question is an excellent bridge to learning about the Definition of Done in general; for example, has the Scrum Team regularly inspected and adapted the Definition of Done?
2. **How would you consider the level of technical debt?** As a new Scrum Master, you want to know everything about the current status of technical debt and dependencies on other Scrum Teams or external suppliers. These issues are directly responsible for your Scrum Team's ability to deliver valuable Increments.
3. **Are you tracking the evolution of technical debt regularly?** I expect an experienced Scrum Team (a) to have a good understanding of the current level of technical debt and (b) to visualize technical debt to ensure its inclusion in any product decisions.

6. Wolpers, "Download the 60 Questions."

4. **How much time per Sprint do you allocate to refactoring and bug fixing?** First, the organization needs to understand the need for refactoring and bug fixing in general. Otherwise, you might be working in a feature factory. Second, as a rule of thumb, I expect the Developers to spend at least 20 percent of their capacity on reducing technical debt and preserving a high-quality customer experience.
5. **How are you sharing knowledge among the Developers?** How are you improving your technical excellence? Developers should be aware of their areas of weakness and have ways that they are working to improve their skills in these areas. They might embrace techniques like pair or mob programming, code mentors, brown bag sessions, hackathons, and the like to improve their development skills.
6. **Do you have any skill gaps requiring formal training classes to improve your team's effectiveness?** This question prepares the Scrum Master for the discussion with the development organization or IT management. For example, the Developers' struggle—such as missed Sprint Goals, a lot of bugs in the production environment, or a substantial lead time to roll out a new functionality—might result from a lack of technical knowledge. The allocation of a training budget to the team might be the way out of the misery.
7. **Do you have the right tools available—software, hardware, cloud applications—to accomplish the team's mission?** This question is self-explanatory but points to a critical impediment the Developers might face: inadequate tools for the job the stakeholders expect them to accomplish. The answer typically provides more food for thought for the new Scrum Master to prepare the discussion with the management.
8. **Are you involved in the hiring process of new Developers?** The Developers shall have the final say over who is joining their team. It defies the basic principle of self-management if outsiders can determine the Scrum Team's composition.
9. **At what stage do you participate in the product discovery process?** It has proven helpful to involve the Developers in the product discovery process as early as possible. The sooner the Developers participate in the process, the less the chances that the Product Owner will pursue solutions that are technically not viable or would not result in a return on investment.
10. **Have you ever interviewed customers of our product to learn more about their needs?** In most cases when Developers haven't yet had direct contact with customers of their product, those Developers are unable to effectively contribute to creating an actionable Product Backlog. This limits the Scrum Team's ability

to achieve their potential. There are often many reasons for this: functional silos within the organization are not collaborating, the Product Owner is not taking collaborative Product Backlog creation seriously, there is no Product Owner in the first place because a steering committee is handling the product design, or the industrial paradigm is still ruling: coders have to code, the business analysts can do the talking, right?

11. **How often are you refining the Product Backlog?** Ideally, the Scrum Team should be continually refining the Product Backlog. If they are not, they should have at least one refinement event per Sprint in which the whole Scrum Team participates.
12. **How many Product Backlog items are you refining during Product Backlog refinement?** Ideally, a team should be refining only work items it can handle within the next two or three Sprints. If they do more than this, they may be wasting their time refining work items they will never work on.
13. **How long does the refinement of a typical work item take?** The refinement should extend only one to two Sprints; context-switching perils and work-in-progress limits also apply to refinement sessions.
14. **How are you creating work items?** There is a tendency for a Product Owner to become more of a technical writer of work items, which the Developers then estimate. Product Backlog item creation should be a joint effort of the whole team to create shared understanding. You want every item in the Product Backlog to be collectively owned by the Scrum Team, not handed down from the Product Owner to the Developers.
15. **How are you discussing work items?** Only during refinement sessions? In collaboration tools? In comments on work items? Every team has its habits, and commenting in tools like Confluence, Jira, Github, or Slack can be an effective means of communication in an organization. So long as the Product Owner agrees, this discussion may occur before the Developers select a Product Backlog item for a Sprint Backlog. Delaying discussion on work items until Sprint Planning can cause problems.
16. **Is your Product Owner changing work items once they become a part of a Sprint Backlog?** And if so, under what circumstances? Making them smaller if the Developers run into a problem is certainly not great, but it is acceptable—if the work item still delivers value. Making it larger after the Sprint Planning, however, is not acceptable. It is paramount that the Product Owner respects the Developers' prerogative to control the composition of the Sprint Backlog.

17. **How are you estimating the effort of a Product Backlog item?** There are plenty of practices on estimating a work item if your Scrum Team estimates at all. (Think of #noestimates, or create similarly sized work items instead.) The emphasis should be on creating a shared understanding among all team members regarding the Product Backlog items. An estimate, the number, is a side effect, not the primary purpose.
18. **Are you estimating in person-hours or story points?** If you believe in estimating, estimating in person-hours is better than not estimating at all. However, I prefer relative estimation with story points, particularly if the application is burdened with legacy code, technical debt, or undone work. In addition, predictability and stakeholder communications become more manageable, as story points have a built-in buffer, covering both effort and complexity.
19. **How do you estimate if the team shares different opinions?** Preferably, you should observe the team's estimation process in real life. But in case you have to ask: is it a typical vote-discuss-revote cycle? Or: when and how do you pick an estimate? (Examples are a 50:50 split [e.g., three times a 3 and three times a 5]—which one do you take? Or a majority split [e.g., two times a 2 and four times a 5]. Or do the estimations cover a range [e.g., from 2 to 8]?) How do you deal with bias—are the juniors following the lead of the seniors? Learning about the estimation process is also an excellent way to learn more about the team-building state.
20. **What are the three main challenges the Developers face today?** This closing question is, by design, an open-ended question to get some ideas for the next Sprint Retrospective.

AGILE TEAM METRICS—THE GOOD, THE BAD, AND THE UGLY

INTRODUCTION

Suitable agile metrics reflect either a team's progress in becoming agile or an organization's progress in becoming a learning organization.

At the team level, qualitative agile metrics often work better than quantitative metrics. At the organizational level, the reverse applies: quantitative agile metrics provide better insights than qualitative ones; see Figure B.1.

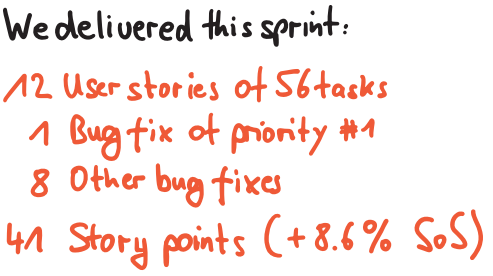

Figure B.1 Agile metrics: No measuring of value created or team effectiveness.

GOOD AGILE METRICS

Good metrics should help us better understand the current situation and gain insight into possible areas for improvement by providing quantitative measures that overcome problems with subjective assessments based on gut feeling that may be biased by emotions or expectations.

A metric should be an indicator for a pattern change, providing an opportunity to analyze potential issues in time. The following three general rules for meaning agile metrics have proven to be helpful:

1. Track only those that apply to the team. Ignore those that measure the individual.
2. Do not measure parameters just because they are easy to track. This practice often results from using various agile project management tools that offer out-of-the-box reports.
3. Record context as well. Data without context—for example, the number of available team members or the intensity of incidents during a Sprint—may be nothing more than noise.

For example, if the (average) sentiment on the technical debt metric is slowly but steadily decreasing, it may indicate that

- A team may have started sacrificing code quality to meet (arbitrary) deadlines,
- A team may not understand what they are doing, or
- A team may have deliberately built some temporary solutions to speed up experimentation.

While the latter probably is a good thing, the first interpretation is worrying. (You should analyze this with the team during a Sprint Retrospective.)

GOOD QUALITATIVE TEAM METRICS: SELF-ASSESSMENT TESTS

Self-assessment tests are well suited if you like to track a team's progress in adopting agile techniques and processes. For example, I use the Scrum Checklist by Henrik Kniberg.[7]

All you have to do is to run the questionnaire every four to six weeks during a retrospective, record the results, and aggregate them; see Figure B.2.

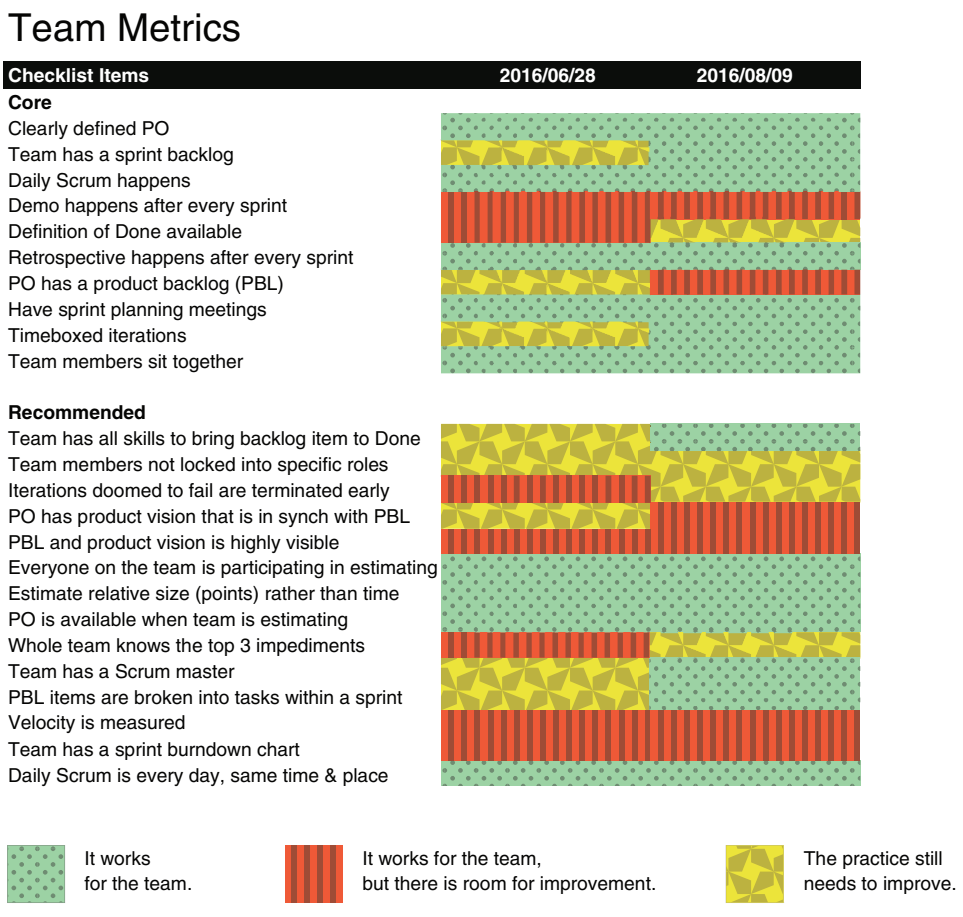
Team Metrics
Checklist Items
2016/06/28
2016/08/09
Core
Clearly defined PO
Team has a sprint backlog
Daily Scrum happens
Demo happens after every sprint
Definition of Done available
Retrospective happens after every sprint
PO has a product backlog (PBL)
Have sprint planning meetings
Timeboxed iterations
Team members sit together
Recommended
Team has all skills to bring backlog item to Done
Team members not locked into specific roles
Iterations doomed to fail are terminated early
PO has product vision that is in synch with PBL
PBL and product vision is highly visible
Everyone on the team is participating in estimating
Estimate relative size (points) rather than time
PO is available when team is estimating
Whole team knows the top 3 impediments
Team has a Scrum master
PBL items are broken into tasks within a sprint
Velocity is measured
Team has a sprint burndown chart
Daily Scrum is every day, same time & place
It works for the team.
It works for the team, but there is room for improvement.
The practice still needs to improve.

Figure B.2 Agile metrics: Team metrics based on Henrik Kniberg's Scrum Checklist.

7. Henrik Kniberg, "The Unofficial Scrum Checklist," April 4, 2021, https://www.crisp.se/wp-content/uploads/2012/05/Scrum-checklist.pdf.

If the resulting Scrum practices map gets greener over time, the Scrum Team is on the right track. Otherwise, you must dig deeper to understand where the team needs to improve. This form of data is valuable input for Sprint Retrospectives and discussions with management, for example, to demonstrate the need for further training.

In addition to this exercise, I also like to run an **anonymous poll** at the end of every Sprint. The vote comprises four questions that are each answered on a scale from 1 to 10:

1. *What value did the team deliver in the last Sprint?* (1 = No value; 10 = Maximum value possible.) See Figure B.3.

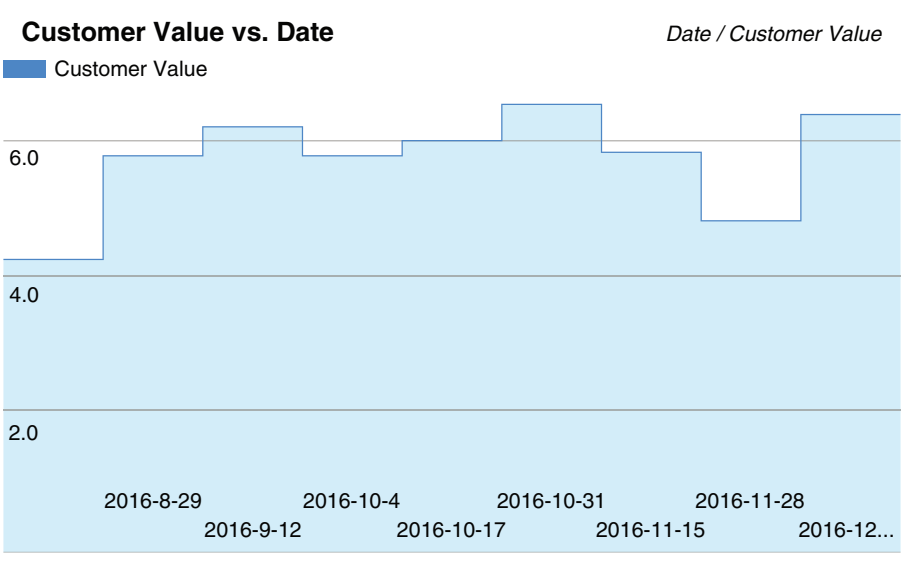

Figure B.3 Agile metrics: Customer Value vs. Date.

2. *How has the level of technical debt developed during the last sprint*? (1 = Rewrite application from scratch; 10 = No technical debt.)
3. *Would you recommend a job opportunity in this organization to a friend with an agile mindset?* (1 = No, I value friendship more than this organization; 10 = Without hesitation.)
4. *Are you happy working with your teammates*? (1: No, I am already looking for a new job; 10: I cannot wait to return to the office on Monday mornings.)

The poll takes less than 60 seconds of each team member's time, and the results are, of course, available to everyone. Again, tracking the development of the four qualitative metrics provides insight into trends that might go unnoticed. Furthermore, the data offer good input for a Sprint Retrospective.

Good Qualitative Team Metrics: The State of Scrum Values

Another good use of a regular anonymous poll is the state of Scrum values within a Scrum Team. Using a Likert scale, you create a poll of five questions covering commitment, courage, focus, openness, and respect (1 might be "The Scrum value is not practiced at all," and 5 might represent "the Scrum Team is perfect at [Scrum value]").

If you run the poll several times and visualize the data with a spider diagram, you can easily track the Scrum Team's progress regarding Scrum values; see Figure B.4.

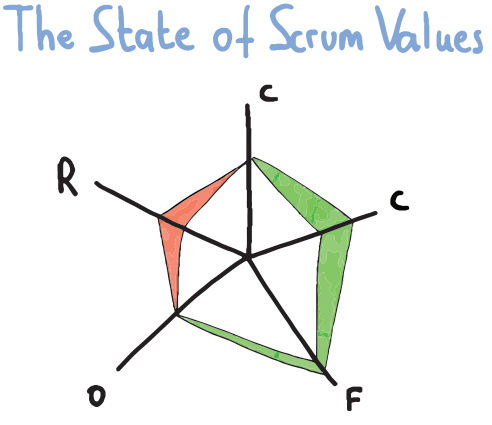

Figure B.4 Agile metrics: The State of Scrum Values.

Good Quantitative Agile Metrics: Lead Time and Cycle Time

Ultimately, any agile transformation's purpose is to create a learning organization. The following metrics apply to the (software) product delivery process but can be adapted to various other processes; see Figure B.5.

In the long run, the transformation may require restructuring the organization from functional silos to cross-functional, self-organizing teams. It will also require analyzing the system itself, for example, determining where unnecessary queues impede value creation.

To identify the existing queues in the product delivery process, measure five dates for each idea:

- The date when a previously validated idea, for example, a user story for a new feature, becomes a Product Backlog item.
- The date when this Product Backlog item becomes part of a Sprint Backlog.
- The date when development starts on this work item.
- The date when the work item meets the team's Definition of Done.
- The date when the Scrum Team releases the newly created Increment to customers.

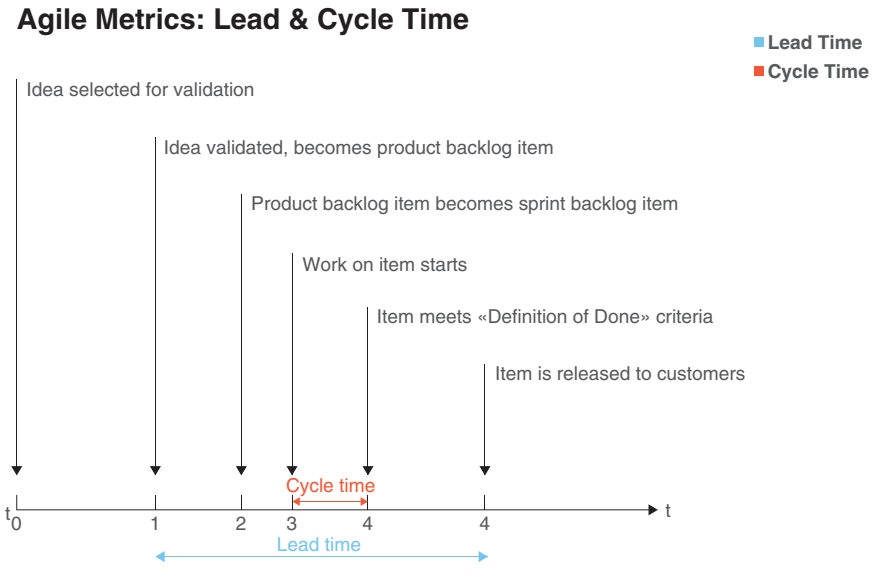

Figure B.5 Agile metrics: Lead Time and Cycle Time.

The **lead time** is the time between the first and the fifth dates, and the **cycle time** is the time between the third and the fourth dates.

The objective is to reduce lead time and cycle time to improve the organization's capability to deliver value to customers. The purpose is accomplished by eliminating dependencies and handovers between teams within the product delivery process.

Here are some helpful practices in this respect:

- Creating cross-functional, empowered Scrum Teams
- Having product teams instead of component teams
- Furthering a holistic, whole-product perspective and systems thinking among all team members

Measuring lead time and cycle time does not require a fancy agile tool or business intelligence software. A simple spreadsheet will do if all teams stick to a simple rule: note the date once you move a ticket. The method even works with index cards.

Figure B.6 compares the median values of lead time and cycle time of three Scrum Teams.

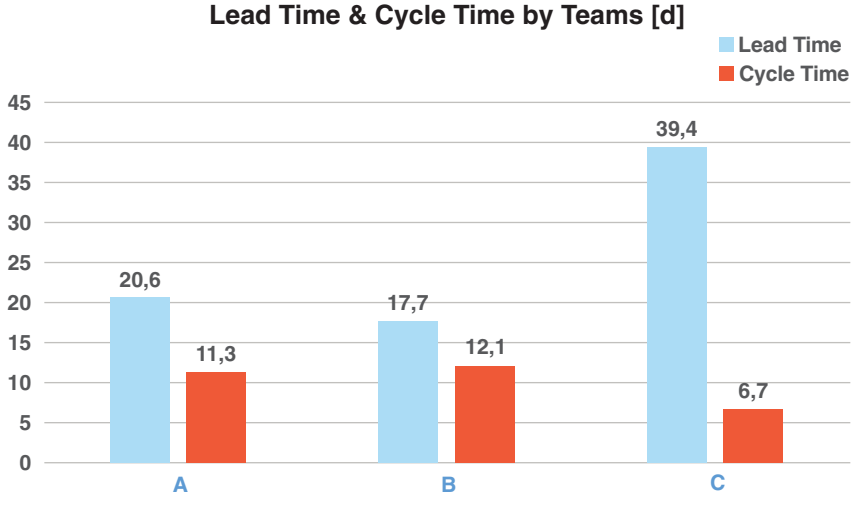

Figure B.6 Agile metrics: Lead Time and Cycle Time by Teams.

The values were derived from analyzing tickets—normal work items and bug tickets—over three months. The Sprint length was two weeks.

OTHER GOOD AGILE METRICS

Esther Derby also suggests measuring the ratio of fixing work to feature work and the number of defects escaping to production.[8]

Another good source for actionable agile metrics is the State of DevOps Report.[9]

BAD AGILE METRICS

A controversial agile metric is team velocity. Unfortunately, team velocity is a notoriously volatile metric and is only usable by an experienced team.

Following are some of the many factors that make even intra-team Sprint comparisons so difficult:

- The team onboards new members
- Veteran team members leave
- Seniority levels of team members change
- The team is working in unchartered territory
- The team is working on legacy code
- The team is running into unexpected technical debt
- Holiday and sick leave reduce capacity during the Sprint
- The canteen serves unhealthy food, causing sick leave
- The team had to deal with serious bugs

You would need to normalize[10] a team's performance each Sprint (which usually is not done) to derive at least some comparable value.

In addition, velocity is a metric that can be easily manipulated. Therefore, I usually include an exercise on how to cook the "agile books" when coaching new teams. Moreover, I have never worked with a team that did not come up with good ideas

8. Esther Derby, "Metrics for Agile," October 11, 2011, https://www.estherderby.com/metrics-for-agile.
9. Google Cloud, "State of DevOps Report," https://cloud.google.com/devops/state-of-devops.
10. Wikipedia, "Normalization (Statistics)," updated September 11, 2022, https://en.wikipedia.org/wiki/Normalization_(statistics).

to ensure that it would meet any reporting requirements based on its velocity. You should not be surprised by this—it is referred to as the Hawthorne effect:[11]

> *The Hawthorne effect (also referred to as the observer effect) is a type of reactivity in which individuals modify or improve an aspect of their behavior in response to their awareness of being observed.*

To make things worse, you cannot compare velocities between different teams because they all estimate differently. This practice is acceptable, of course, as estimates are usually not pursued merely for reporting purposes. Instead, estimates are no more than a side effect of the attempt to create a shared understanding among team members on a work item's why, how, and what.

Ugly Agile Metrics

So far, the ugliest agile metric I have encountered is story points per developer per time interval. This metric is equivalent to measuring the lines of code per developer or hours worked per developer. The metric is useless, as it doesn't provide any context for interpretation or comparison.

Equally useless agile metrics are, for example, the number of certified team members or the number of team members that accomplished workshops on agile practices.

Conclusion

A Scrum Team does not need to have a complete set of metrics on their first day of work.

If you can record only a few data points, measure lead time and cycle time to help you to see bottlenecks in the flow of work. You may also consider tracking an individual team's fluency in Scrum by measuring qualitative signals, for example, based on self-assessment tests like the Scrum test by Henrik Kniberg.

However, by all means, agree as a Scrum Team on a set of metrics that can provide insight into your progress in improving and in improving and tracking yourself. Creating a shared understanding by including all team members is critical to picking the right metrics. Moreover, it is essential for all the team members'

11. Wikipedia, "Hawthorne Effect," updated October 2, 2023, https://en.wikipedia.org/wiki/Hawthorne_effect.

acceptance of change if the metrics point to what is necessary to improve as a Scrum Team.

Finally, do not use prefabricated reports in ticketing systems just because they are readily available. Identify what metrics will support your Scrum Team and then choose a tool if it helps you collect and report on those metrics.

BUILDING COMMUNITIES OF PRACTICE

INTRODUCTION

Creating an Agile community of practice—or CoP, for short—helps win hearts and minds within the organization as it provides to the agile transformation.

Read more to learn how to get your Agile community going, even without a dedicated budget, and how to make it work with distributed teams.

HOW TO BECOME AN AGILE ORGANIZATION—WHEN THE PLAN MEETS REALITY

Typically, the recipe for becoming an agile organization goes like this: You need the commitment from the C-level to change the organization's culture and, thus, its trajectory. It would help if you also had strong support from the people in the trenches who want to become autonomous, improve their mastery, and serve a purpose. Then—in a concerted effort—the top and the bottom of the hierarchy can motivate the middle management to turn into servant leaders (or perhaps follow Haier).[12]

12. Haier's RenDanHeYi model is an innovative organizational approach emphasizing decentralization and entrepreneurship. The term is derived from Chinese; *Ren* refers to employees, *Dan* stands for user value, and *HeYi* signifies their integration.

    This model dismantles traditional organizational hierarchies, emphasizing eliminating middle management. Instead, the organization is made up of many small, self-managed units, or microenterprises, that operate like independent startups. Each unit is directly linked to customer needs, fostering an entrepreneurial spirit and improving response time to market changes.

    Middle management, in the traditional sense, does not exist. Everyone becomes an entrepreneur or owner, taking responsibility for their microenterprise's success or failure. While empowering employees, this paradigm shift also requires them to upskill and take on leadership roles, fostering innovation and customer-centricity.

    While RenDanHeYi is reportedly successful in Haier, it requires a significant shift in mindset and might suit only some organizations or cultures. Learn more: Bill Fischer, Umberto Lago, and Fang Liu, *Reinventing Giants: How Chinese Global Competitor Haier Has Changed the Way Big Companies Transform* (Jossey-Bass, 2013).

Accordingly, an action plan often starts with hiring a consultancy to help figure out a more actionable roll-out plan, mainly comprising training and workshops, initial team-building activities, and possibly some audits concerning financial reporting requirements, technology, or governance issues.

What this kind of orchestrated initiative often neglects is the grassroots part of any successful change: providing room and resources to the members of the organizations to engage in a self-directed way with the change process itself.

The Purpose of an Agile Community of Practice

The purpose of an Agile community of practice has two dimensions:

- Internally, it provides an educational capacity for agile practitioners and change agents. There is no need to reinvent the wheel at the team level; regularly sharing what has been successful or a failure in the context of the transition will significantly ease the burden of learning.
- Externally, the Agile community of practice contributes to selling, for example, Scrum to the rest of the organization by informing and educating its members. The members of the Agile community also serve as the first servant leaders and thus as role models for what becoming an agile organization will mean in practice. They bring authenticity to the endeavor.

Winning hearts and minds by being supportive and acting as a good example day in and day out is a laborious and less glamorous task. It requires persistence–and being prepared not to take no for an answer but to try again. Reaching the tipping point of the agile transition will likely be a slow undertaking with few signs of progress in the beginning. Moreover, management tends to underestimate the inherent latency. According to a study directed at the path of acceptance of new social norms, the tipping point for social change in an organization is around 25 percent.[13] It shows that a committed minority can have a lasting effect once it attracts others willing to join the cause.

13. The study found that when 25 percent of people in a group adopt a new social norm, it creates a tipping point where the entire group follows, suggesting that a small minority can influence the majority to adopt a new social standard. Damon Centola, Joshua Becker, Devon Brackbill, and Andrea Baronchelli, "Experimental Evidence for Tipping Points in Social Convention," *Science* 360, no. 6393 (2018): 1116–19, https://doi.org/10.1126/science.aas8827.

A PORTFOLIO OF SERVICES AND OFFERINGS OF AN AGILE COMMUNITY OF PRACTICE

INTERNAL OFFERINGS, SERVING THE COMMUNITY

Improving the level of mastery of the members of the Agile community of practice is not difficult. My top picks are as follows:

- **Sharing is caring:** Hoarding information is one of the worst anti-patterns of an agile practitioner. Hence sharing everything, for example, from Retrospective exercises to information resources (newsletter, blog posts, etc.) to working material and supplies, is helpful. A wiki might be the right place to start.
- **Training and education, part 1:** Organize regular workshops among the agile practitioners to train each other. If only some people are co-located, either record the training for later use or create live virtual classes. Using Liberating Structures ensures that everyone will be included and given a voice.
- **Training and education, part 2:** If you have a budget available, invite the leading voices of your field to the organization to train the trainers. You cannot overestimate the psychological impact of such training on the change agents. It is proof that change is what the organization's leadership wants.
- **Organize events:** Have regular monthly (virtual) events for the agile practitioners and others from the organization, and host meetups with external speakers. Make sure that all practitioners meet in person at least once a quarter for a day-long mini-conference. If in-person meetings aren't feasible, host a virtual conference instead.
- **The annual conference:** Consider hosting an organization-wide yearly State of Agile conference to share lessons learned, success stories, and failures. Again, this can be a virtual conference as well.
- **Communication:** Use a direct messaging application to foster frictionless communication among the community members.
- **Procurement:** Find a workaround to allow unlisted suppliers to provide supplies, such as special pens or stickies. It also could be that required software for remote work is currently unavailable through official channels. (Perhaps there is a freelancer or contractor among the practitioners who can help with that.)

External Offerings, Serving the Organization

Generally, what is working for the Agile community of practices is also suitable for the organization's members, probably with a different focus. Try, for example, the following:

- **Provide training:** Provide hands-on training classes in close collaboration with the change agents. Consider a less demanding format than a typical day-long training class—a focused one-hour class in the late afternoon may prove to be just the right format for your organization. (Tip: Avoid the necessity for participants to apply somewhere to be allowed to the class. Omitting an application process will massively improve attendance rates.) Also, consider offering a kind of curriculum that is comprised of several of those lightweight classes. This format is also perfectly suited for live virtual classes.
- **Communication:** Consider running a website or blog beyond the Agile community of practice's wiki to promote the organization's path to becoming an agile organization. So far, the best means I have encountered to foster engagement among change agents and early adopters is a weekly or biweekly newsletter within the organization.
- **Make Agile mandatory for new colleagues:** Educate all new hires on agile principles and practices to support the company's repositioning. I described this approach in detail in "App Prototyping with Absolute Beginners."[14]
- **Gain visibility:** Selling Agile to the organization to win hearts and minds is best achieved by making Agile tangible at a low-risk level for the individual. For example, organize Lean Coffee sessions or Knowledge Cafés regularly, thus providing a safe environment to check out this Agile thingy. Invite people directly to ceremonies, such as Sprint Reviews—if you practice Scrum—that might be of interest to them. (Guerilla advertising is welcome: I used to print invitations for Sprint Reviews of my team, put them inside elevators, or leave them on the cafeteria tables.)
- **Provide access to change agents and Scrum Masters:** Why not offer an informal way of contacting agile coaches and change agents? Some people shy away from asking supposedly stupid questions in the open and may be hard to reach otherwise.

14. Stefan Wolpers, "App Prototyping with Absolute Beginners—Creating a Shared Understanding of How Empiricism Works," April 4, 2021, https://age-of-product.com/app-prototyping-with-absolute-beginners-agile-experiments.

- **Provide transparency:** Occupy a space at a highly frequented part of a building or the campus to show what Agile is about and provide an overview of practices, courses, regular events, and so on.
- **Host events:** Try to organize regular events for the organization, for example, by providing lessons learned from teams spearheading the agile transition. Create a schedule in advance and stick to it. Perseverance is critical to fighting the notion that becoming agile is merely a management fad that will disappear soon. But, again, this works perfectly well with distributed teams.

Overcoming Resistance to an Agile Community of Practice

I have yet to witness open pushback from an organization about creating an Agile community of practice. More likely, you will encounter complacency or ignorance at the management level. Sometimes, the budgeting process will be utilized—willingly or not—to impede the creation of a community.

But even when you are financially restrained, there is still enough room to move the Agile community of practice ahead. There are several free services available that provide video conferencing and hosting, blogs, event organization, or newsletter services. In my experience, it is less a question of available funds, and rather more that you need to overcome your anxiety and get going without waiting for written approval from whomever. Assuming responsibility as an agile practitioner in general and a Scrum Master in particular by starting a community and moving the transition forward sounds like authentic servant leadership.

Conclusion

Creating an agile community of practice is vital to becoming an agile organization. It provides a lot of the groundwork necessary to convince the organization members that becoming agile is neither hazardous nor a fad but a trend and, thus, an excellent chance for everyone involved.

CREATING PRODUCT GOALS AS A COLLABORATIVE PRACTICE

INTRODUCTION

> *The Product Goal describes a future state of the product which can serve as a target for the Scrum Team to plan against.*[15]

While the Product Owner is responsible for creating and communicating the Product Goal, it is beneficial to include the rest of the Scrum Team in the process, similar to other Product Backlog management tasks. This collaborative approach, like many others, aims to foster a shared understanding among all Scrum Team members about the why, what, and how of the team's journey. Also, it reduces the possibility of creating technical risk due to a lack of understanding on the Product Owner's side.

In this regard, Ralph Jocham developed and shared a valuable alignment tool: the Product Goal Canvas; see Figure B.7.[16]

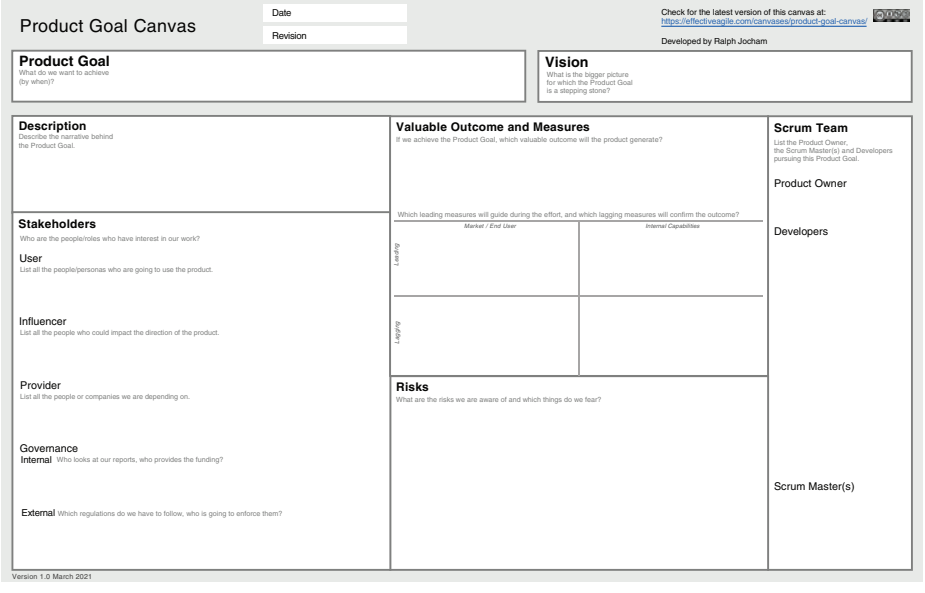
Product Goal Canvas
Date
Revision
Check for the latest version of this canvas at:
https://effectiveagile.com/canvases/product-goal-canvas/
Developed by Ralph Jocham
Product Goal
What do we want to achieve (by when)?
Vision
What is the bigger picture for which the Product Goal is a stepping stone?
Description
Describe the narrative behind the Product Goal.
Valuable Outcome and Measures
If we achieve the Product Goal, which valuable outcome will the product generate?
Which leading measures will guide during the effort, and which lagging measures will confirm the outcome?
Market / End User
Internal Capabilities
Leading
Lagging
Scrum Team
List the Product Owner, the Scrum Master(s) and Developers pursuing this Product Goal.
Product Owner
Developers
Scrum Master(s)
Stakeholders
Who are the people/roles who have interest in our work?
User
List all the people/personas who are going to use the product.
Influencer
List all the people who could impact the direction of the product.
Provider
List all the people or companies we are depending on.
Governance
Internal Who looks at our reports, who provides the funding?
External Which regulations do we have to follow, who is going to enforce them?
Risks
What are the risks we are aware of and which things do we fear?
Version 1.0 March 2021

Figure B.7 The Product Goal Canvas by Ralph Jocham.[17]

15. Ken Schwaber and Jeff Sutherland, "Product Backlog," The 2020 Scrum Guide™, 2020, https://scrumguides.org/scrum-guide.html#product-backlog.
16. Ralph Jocham, Product Goal Canvas, March 2021, https://effectiveagile.com/canvases/product-goal-canvas.
17. Creative Commons, Attribution-NonCommercial-ShareAlike 2.0 Generic (CC BY-NC-SA 2.0), https://creativecommons.org/licenses/by-nc-sa/2.0/.

THE PRODUCT GOAL CANVAS

The canvas comprises seven areas:

- **Vision:** In the context of our product's big picture, the Product Goal Canvas bridges the product vision at the strategic level to the Product Backlog at the tactical level.
- **Product Goal:** What higher-level outcome shall the Scrum Team accomplish over the next several Sprints?
- **Description:** Tell a story about why the outcome would be valuable to customers and your organization.
- **Valuable outcome and measures:** Describe the outcome, what we will observe, and how it will create value for our customers. How do we know we are on track to accomplish the Product Goal? Moreover, how can we confirm that the planned outcome has been achieved?
- **Stakeholders:** Who are the people we need to communicate and collaborate with to accomplish our Product Goal?
- **Risks:** What risks are we facing that might prevent us from accomplishing the Product Goal? What would we consider a failure, and how would it materialize?
- **Scrum Team:** Who will be working on realizing the Product Goal?

Depending on your preliminary work, the time to create the first version of the canvas will vary. For example, if you have already invested in maintaining a Business Model Canvas[18] and know your stakeholders well, it may take 60 to 90 minutes. On the other hand, it will take much longer if you haven't done any preliminary work on the product and business side.

Once you create the first version, please regularly inspect and adapt your Product Goal canvas for as long as you work on the respective Product Goal.

DAILY SCRUM WITH DISTRIBUTED TEAMS

VIRTUAL LIBERATING STRUCTURES FOR A REMOTE DAILY SCRUM

Contrary to misinformed belief, the Daily Scrum's 15-minute timebox is not intended to solve all the issues raised during the event. The purpose of the Daily

18. Strategyzer AG, "Business Model Canvas," https://www.strategyzer.com/canvas/business-model-canvas.

Scrum is to create transparency that triggers further inspection of the issue. This inspection and the related response may occur outside of the Daily Scrum event. For example, if the Developers need to adapt the Sprint Plan or the Sprint Backlog, they are free to do so at any time. In my experience, most Daily Scrum issues result from misunderstanding this core principle. We hence look at the Daily Scrum's two parts separately.

Virtual Daily Scrum, Part 1: Are We Still on the Right Track to Accomplish the Sprint Goal?

Part 1 is the 15-minute timebox where the Developers inspect the progress toward achieving the Sprint Goal since the last Daily Scrum: Have we learned something since yesterday that should force us to reconsider our current plan?

One way of applying this pattern to a remote Daily Scrum is to run a virtual Mad Tea[19] to rapidly collect and distribute information from all Developers among the team members. Of course, we cannot create two concentric circles of attendees facing each other. However, we can use the set of questions to create a prompt—a half-sentence that the team members have to complete—and the chat channel to establish a quick picture of the team's sentiment.

As the moderator, prepare the necessary prompts regarding the team's progress toward the Sprint Goal, for example, "Today, I will support my team by _____." Then post that prompt to the chat and ask the participants to add their answer(s) but not to hit Enter yet. That is done simultaneously by all attendees when the moderator asks for it. This way, the Developers can generate insight synchronously at a rapid pace, identifying potential areas where more collaboration is needed to ensure that the Scrum Team is meeting the Sprint Goal.

Virtual Daily Scrum, Part 2: What I Need from You

Let us move on to the part of the remote Daily Scrum, where the Developers collaborate on solving issues that might impede their ability to achieve the Sprint Goal. Consider the following microstructures for the second phase of the remote Daily

19. Henri Lipmanowicz and Keith McCandless, "Mad Tea (v1.2)," Liberating Structures, https://www.liberatingstructures.com/mad-tea.

Scrum, when Developers may need individual support or where the whole team needs to decide how to adapt in the light of new learnings:

Troika Consulting[20] proves to be especially helpful, for example, at sourcing support at the individual level from other Developers on how to solve a technical issue. To run it in a distributed team, we create breakout rooms for groups of three. First, the consultants and the consultee have the initial conversation, where the consultee introduces their challenge, and the consultants ask clarifying questions. Then, the consultee turns around on their chairs for the consulting phase. Alternatively, both consultants stop broadcasting their video streams, so the consultee listens to what they have to say. In the final round, the consultee provides consultants feedback on their efforts.

Min Specs[21] has proven to be a very effective virtual Liberating Structures microstructure when it becomes apparent that the original plan to achieve the Sprint Goal is no longer valid and needs to be trimmed, probably in collaboration with the Product Owner. Like **What, So What, Now What?**,[22] a remote Min Specs is a sequence of individual work and groupwork based on breakout rooms, aggregating findings in shared workspaces to be shared with the whole Development Team in the end.

What I Need From You (**WINFY**)[23] could offer the right frame to articulate core needs to achieve the Sprint Goal and coordinate and involve everyone in the collective effort when the team is relatively inexperienced. (Other cases comprise structures centered around component teams or challenging software architectures.) Again, we employ breakout rooms, chat, and shared workspaces in a remote setting. Embedded **1-2-4-All**[24] or **What, So What, Now What?** may also support the effort.

TRIZ[25] may prove helpful when the team needs to reaffirm its plan to achieve the Sprint Goal, for example, because of outside pressure. TRIZ combines essential elements of virtual Liberating Structures: breakout rooms, embedded 1-2-4-All, and joined workspaces. As a Scrum Master, you may consider supporting the effort by time-keeping on behalf of the participants, as they are likely to lose track of time.

20. Henri Lipmanowicz and Keith McCandless, *The Surprising Powers of Liberating Structures* (Liberating Structures Press, 2013): 194–96.
21. Lipmanowicz and McCandless, *The Surprising Powers,* 228–31.
22. Ibid., 197–201.
23. Ibid., 266–70.
24. Ibid., 167–71.
25. Ibid., 187–90.

Another approach to solving the aforementioned challenge is the **Discovery and Action Dialog (DAD)**[26] to "Discover, Invent, and Unleash Local Solutions to Chronic Problems." A remote setup may result in a high bias for action, particularly on the side of stakeholders and the management, due to a perceived loss of control, impeding a Scrum Team from achieving its Sprint Goal. DAD can support the Developers in discovering outside intervention and how to address the situation by identifying positive deviant (PD) behaviors and practices to provide solutions to real (and perceived) problems. Technically, a virtual DAD applies standard procedures like breakout rooms, embedded **1-2-4-All**, joined workspaces, and sometimes a **Shift & Share**.

Supporting Liberating Structures

Two Liberating Structures microstructures can support a remote Daily Scrum and particularly its second phase: **1-2-4-All** and **Lean Coffee.**[27]

To cover **1-2-4-All**, we need breakout rooms and a place to aggregate the findings. We start with everyone in the group for a minute in silence; then, we split the whole group into pairs using Zoom's or Team's breakroom feature for 2 minutes. After that round, we merge two pairs into a group of 4 for 5 minutes—this has to be done manually by the host—and the group aggregates its findings, for example, on a Google sheet prepared for each group in advance. Finally, we can introduce each group's findings to the whole group by screen sharing in a **Shift & Share**.

Lean Coffee is an excellent example of a workaround for virtual Liberating Structures. Gather all the input in the usual way, for example, engaging in 1-2-4-All, and post it on an online whiteboard while voting is turned off. (Use several columns if the whole group is large to speed up the gathering process.) Then ask the entire group to cluster similar topics. Once the clustering is done, turn on the voting and order the remaining entries by votes. From here, continue with a group discussion or engage smaller groups with breakout rooms.

Conclusion

Daily Scrum events—remote or not—are essential to a team's self-management and, consequently, its ability to achieve the Sprint Goal. While the first phase, the 15-minute timebox, is relatively straightforward, the second, the problem-solving

26. Lipmanowicz and McCandless, 202–7.
27. Lean Coffee, "Lean Coffee Lives Here," http://leancoffee.org.

phase, requires more effort on the side of the moderator, the Scrum Master, or Developers in a remote setup. Nevertheless, given the right training and support level, a distributed Scrum Team can also fully embrace the second phase.

How to Create a Definition of Done

As outlined before, creating the Definition of Done shall be a collaborative exercise of the whole Scrum Team, possibly even including some of the stakeholders at a later stage.

My practice in creating a Definition of Done is to analyze and reflect on earlier experiences with a Definition of Done: What made a Definition of Done work for previous Scrum Teams? Here, I like to use a Liberating Structure called *Appreciative Interviews*:[28]

1. In pairs, decide who goes first. Next, that person shares a story of when they were part of a successful Scrum Team and how the Definition of Done contributed to the success. What happened? What made the success possible? The other person channels their inner journalist by asking clarifying questions and taking notes. Switch roles after 5 minutes.
2. In groups of four, retell the story of your respective partner. The other group members listen for patterns that connect the stories. Switch roles after 2 minutes.
3. In the whole group, identify the most helpful Definition of Done patterns.

Based on these patterns of a helpful Definition of Done, we now use another Liberating Structure—1-2-4-All—to source concrete suggestions on the team's first version of the Definition of Done.

In the first round, everyone individually and in silence makes suggestions, maybe five to ten. In the second round, two team members compare their proposals, find new ones, and merge those into another set of five to ten suggestions. In the third round, the step is repeated, with two pairs comparing and merging their two sets of suggestions. Finally, in the last round, the team comes together to share their findings. You can use another Liberating Structure—Shift & Share[29]—for that purpose.

28. Lipmanowicz and McCandless, *The Surprising Powers,* 182–86.
29. Lipmanowicz and McCandless, 212–16.

Now, the team has a set of suggestions for the new Definition of Done. The next step is to cluster the suggestions based on topics and identify those that the Scrum Team considers helpful to become a part of the first version of the Definition of Done. A useful exercise is, for example, a practice called White Elephant.[30]

Here, the team distributes all suggestions along a scale ranging from "mostly useless" to "most useful." Every suggestion is individually placed by a team member, explaining why they chose the respective spot. Once the team has placed all proposals, every team member can move suggestions while sharing an explanation on why they believe the new placement of the suggestion for the Definition of Done offers more value to the Scrum Team. If a team member agrees with the positioning of all placements, they pass. This adjustment process is continued until everyone passes on repositioning any suggestions.

The last step is to agree on which suggestions to take into the new Definition of Done, starting from "most useful" and moving down the scale. The result is the Scrum Team's first Definition of Done.

I recommend starting with a lightweight Definition of Done and revisiting it regularly to determine whether it still represents the desired quality level. I suggest doing so in the beginning in shorter intervals, maybe every two to three Sprints. The right place for this exercise is the Sprint Retrospective.

Meta-Retrospectives—How to Get Customers and Stakeholders Onboard

Introduction

A Meta-Retrospective is an excellent exercise to foster collaboration within the extended team, create a shared understanding of the big picture, and immediately create valuable action items. It comprises the team members of one or several product teams—or representatives from those—and the stakeholders. Participants from the stakeholder side are people from the business as well as customers.

30. Glaudia Califano, "Trumpeting For All Voices to Be Heard with White Elephant—Facilitating for Participation," May 11, 2022, https://www.scrum.org/resources/blog/trumpeting-all-voices-be-heard-white-elephant-facilitating-participation works for me.

Meta-Retrospectives are useful both as a regular event, say once a quarter, or after achieving a particular milestone, for example, a specific release of the product.

How to Run a Meta-Retrospective

The Meta-Retrospective format I describe here is based on Zach Bonaker's WADE-matrix,[31] extended by an additional practice at the beginning of the retrospective. To frame the level of (necessary) openness of the upcoming conversation, I run a short exercise bringing the Scrum values back into the hearts and minds of the attendees. After all, we are organizing the Meta-Retrospective to also address the elephants in the room.

The Meta-Retrospective itself does not require any knowledge of agile practices and is hence suited for practically everyone. This format can easily handle 15 or more people, provided the room is large enough that people can get together for discussions. Also, you need at least one large whiteboard in the room, as most of the work will happen initially on this wall. Note that Meta-Retrospectives can also be held in a virtual setting.

The Scrum Values Exercise

Running the Scrum Values exercise is simple:

- Ask the participants to pair up and identify their choice of the three most essential traits that will support collaborating as a team within 3 minutes. (It's helpful to provide an example of what is not helpful, such as yelling or pointing fingers.)
- Then ask every pair to introduce their choices to the rest of the attendees and put them on the whiteboard. If similar traits are already available, they can be clustered.
- Once all stickies are on the whiteboard, step forward and explain what Scrum Values are about and why they are helping to guide a team to accomplish its task. (Ensure that you mention that challenging issues must be addressed in a civilized manner.)

31. Zach Bonaker, "Seeing the System with the WADE Matrix," July 3, 2019, https://www.scatterspoke.com/post/seeing-the-system-with-the-wade-matrix.

- Then put five stickies with the Scrum values written on them—courage, focus, commitment, respect, and openness—on the whiteboard and ask the attendees to align their findings with the Scrum values.
- Once that is done, you are good to go with the Meta-Retrospective.

THE META-RETROSPECTIVE EXERCISE

Start the Meta-Retrospective by drawing the first axis onto the whiteboard and note that the axis represents a continuum. Then ask the attendees to pair up again but choose a different partner; see Figure B.8.

Now ask them to pick their three most important learnings looking back. Timebox this creation phase to 3 to 5 minutes. After the stickies with the learnings are available, ask every pair to introduce them to the rest of the attendees and put them on the whiteboard. (Again, cluster stickies where appropriate.)

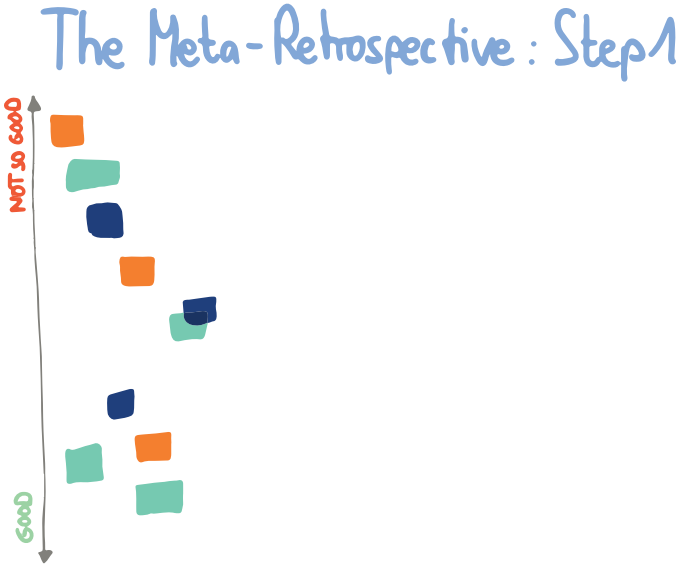

Figure B.8 Create the vertical axis labeled "Good" to "Not so good."

In the next step, introduce the second axis—the "influence" axis—another continuum; see Figure B.9.

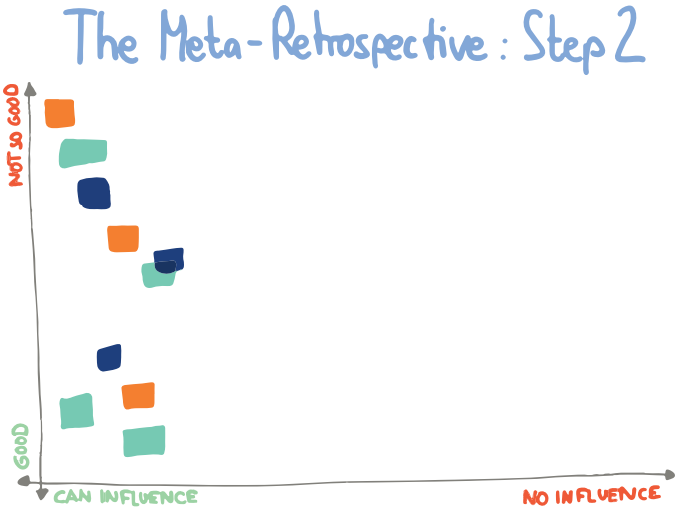

Figure B.9 Introduce the horizontal axis labeled "Can influence" to "No influence."

Then ask the participants to align all stickies on the whiteboard with the second axis. You can stop this part of the exercise once the participants stop moving stickies on the whiteboard; see Figure B.10.

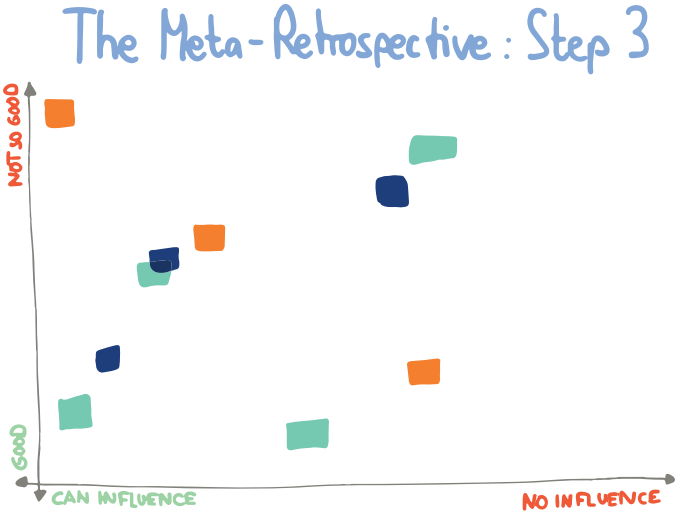

Figure B.10 Now align all stickies with the second axis to create a two-dimensional area.

Now it is time to turn the pattern into a 2 × 2 matrix and label the four quadrants accordingly (see Figure B.11):

1. *Get to Work*—This is the area of immediate impact.
2. *Talk to the Management*—These issues are impeding you; escalate them to the management.
3. *Luck*—That went well, but do not invest any effort here.
4. *Keep Doing*—Nothing to change here at the moment.

For the last step, focus on the upper-left quadrant—"Get to Work"—and ignore the bottom two quadrants. Probably, there will also be time to address the upper-right quadrant ("Talk to the Management"). Start by moving the stickies from the upper-left quadrant to a different part of the whiteboard, and prepare them for a dot-voting to figure out the ranking of the issues; see Figure B.12. (I usually issue three to five dots to each attendee for this purpose. The voting may take up to 5 minutes.)

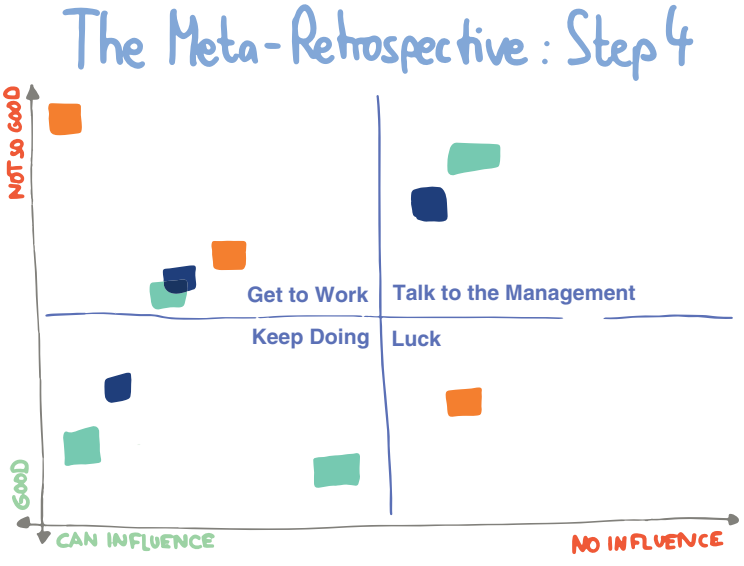

Figure B.11 Turn the pattern into a 2 × 2 matrix, and label the four quadrants.

Once the attendees accomplish the voting, generate some action items by running a Lean Coffee-style discussion based on the ranked issues.

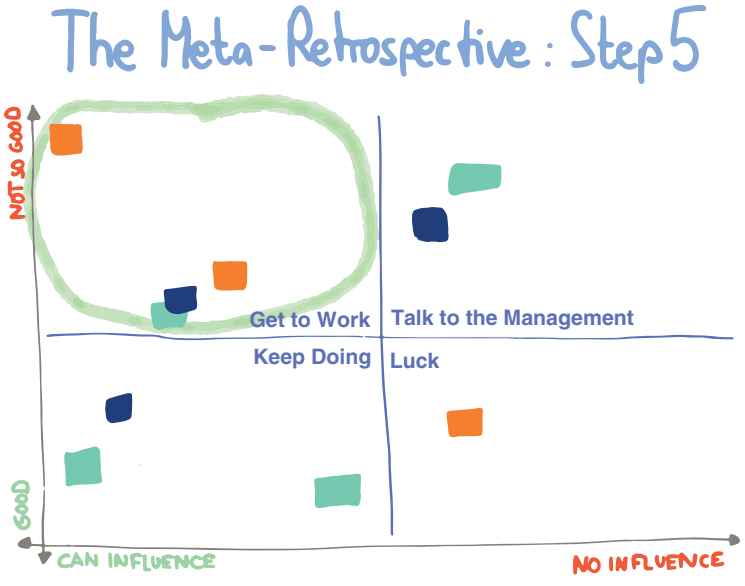

Figure B.12 Delve into the issues of the upper-left quadrant with a Lean Coffee.

CONCLUSION

Running a Meta-Retrospective is an excellent exercise to foster collaboration within the extended team, create a shared understanding of the big picture, and immediately create valuable action items. Best of all, it takes less than two hours to make the ideas of avoiding Muda[32] and practicing Kaizen[33] tangible to everyone.

PEER RECRUITING

INTRODUCTION

Peer Recruiting is a practice in which the team conducts recruiting activities and makes hiring decisions. It is based on the idea that the team best understands the

32. Muda, a Japanese term from the Toyota Production System, refers to activities that add no value to the final product, essentially wasteful actions. For more, check James P. Womack and Daniel T. Jones, *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*, 2nd ed. (Free Press, 2003).
33. Kaizen, a Japanese term meaning "change for the better," embodies the philosophy of continuous improvement in work processes and efficiency. For more insight, see Masaaki Imai, *Kaizen: The Key to Japan's Competitive Success* (McGraw Hill Education, 1986).

new skills that it needs and what sort of person will best be able to work with other team members; see Figure B.13.

The following guide to peer recruiting is based on my experience recruiting Scrum Team members over the last 15-plus years.

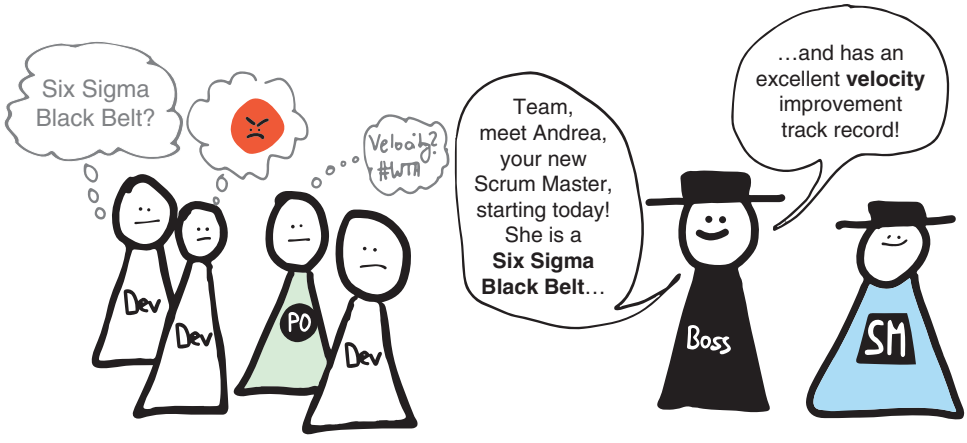

Figure B.13 When managers make hiring decisions for the team, they disrespect and demotivate the team.

PEER RECRUITING AND THE NEW ROLE OF PEOPLE OPERATIONS

Peer recruiting does not imply that People Operations will be rendered obsolete. On the contrary, they will continue being a significant contributor to the success of the whole organization. However, People Operations will no longer choose someone from the candidate pool and present that individual to the Scrum Team as the new teammate. Instead, they will support the team in picking the "right" candidate and ensuring that the legal and administrative side is taken into account.

Typical tasks of the peer recruiting process that People Operations will provide to a Scrum Team comprise the following:

- Creating the remuneration package for the position in question (in compliance with the organization's principles)
- Handling contractual and administrative issues (social security, visas, work permits, etc.)

- Supporting the Scrum Team in creating a job advertisement (if required)
- Placing job advertisements and running corresponding campaigns
- Doing background checks and prescreenings of applicants
- Organizing interviews and trial days (from travel arrangements and meet and greet to introducing the organization)
- Collecting the Scrum Team's feedback after interviews or trial days
- Handling the signing of the contract
- Finally, kicking off the onboarding process for the new teammate

These steps hold a significant opportunity for People Operations to become a change agent for the organization, contributing to its agile transition by ensuring that new hires will have the required agile mindset.

WHY BOTHER WITH THE INCLUSION OF THE SCRUM TEAM AT ALL?

Why will a change of process be required in the first place? There are plenty of reasons; my top three are:

1. It's consequent. On the one hand, the Scrum Team is empowered to make decisions that directly impact the return on (product) investment. On the other, are they being patronized by deciding on new teammates for them?
2. It means the team has skin in the game. The team members will be motivated to go the extra mile to make the new connection work. Now, it is their responsibility, too.
3. Not involving the Scrum Team immediately signals to all candidates that your organization isn't agile but merely "doing Agile"—a weak value proposition in the war for talent with an agile mindset.

THE EIGHT-STEP PEER RECRUITING PROCESS OF HIRING A SCRUM MASTER

Now, let's have a detailed look at the proposed process, which has been proven to be motivating and successful several times in different organizations.

1. What Kind of Scrum Master Are You Hiring?

This is the question you will need to answer in the first place: What is the purpose of building autonomous teams in your organization? Does the organization want to become (or stay) agile? Or is the organization just "doing Agile"?

You should always hire for the mindset in the "team of teams" universe. While you can quickly teach skills, training an unsuitable candidate in the right mindset will be futile most of the time. Therefore, the following description targets organizations that want to become agile.

2. Create a Job Advertisement

In an ideal world, running a job ad wouldn't be necessary. Instead, someone in the product development organization would personally know a suitable individual and introduce them to the team (and the organization).

Unfortunately, genuinely agile people are in short supply, so there will likely be the need to create a job ad for the website and other channels.

At this point, I recommend kicking off the collaboration between the Scrum Team and People Organization. Most ads that People Operations departments produce for agile jobs are simply awful. Their usual "I don't know what this is all about, but I have to come up with an ad by noon, so I copied the text from a competitor" approach scares away suitable candidates because they sense the lack of competence.

Instead, I suggest sitting with the whole team, sharing a coffee, and getting the ad right by being authentic and human and reflecting the organization's culture.

3. Run the Job Advertisement

Running the ad is the People Operations department's job. However, the Scrum Team may have good suggestions of places to look, such as social media channels, professional networks, or meetups of the local Agile community.

4. Prescreening Applicants

It would be helpful for the team if People Operations prescreens applicants. For example, this screening could be the standard background check or a first analysis if a candidate is suitable for legal reasons or in compliance with the hiring organization's internal programs, such as diversity initiatives.

Unsuitable candidates should then be flagged and removed from the pool. (Although the Scrum Team should be aware of that for transparency reasons.)

5. Discuss Suitable Applicants with the Team

The team members should then be provided with access to all suitable candidates, preferably in the form of anonymized CVs: no photos, no age, no gender, no ethnic group, no religious beliefs—any information on candidates that might trigger a bias of whatever kind should be excluded.

Also, you will normalize information on a candidate's public profile (e.g., blog, Twitter, or other accounts). For example, a summary such as "A is actively contributing to the Scrum community by running a meetup as well as creating a newsletter with 800 subscribers" will suffice for the selection process.

If this approach requires scissors, glue, and Xerox machines, so be it. However, please remember that these biases are triggered on autopilot and that there is no willpower known to humanity that could prevent biases from interfering with the selection process.

Then have a joined meeting—People Operations and all Scrum Team members—and discuss whom to invite for the in-person interviews.[34] (A simple dot-voting will suffice in the end.)

6. Running the Interviews

The *Hiring: 73 Scrum Master Interview Questions to Identify Suitable Candidates* e-book[35] provides a broad set of questions (and possible answers) spanning several categories.

Those are the starting point for the peer recruiting interviews. The interviews aim to identify those who will be invited for a trial day with the Scrum Team.

Instead of one or two team members each having an extended interview with a candidate, I recommend running interviews of 30 minutes each with as many team members as possible. The trick is that you split the questionnaire evenly among the interviewers and later aggregate the answers. Thus, you will obtain more constructive feedback from all the interviewers.

34. Don't forget to consider this investment in the team's future capabilities during Sprint Planning.
35. Stefan Wolpers, *73 Scrum Master Interview Questions to Identify Suitable Candidates,* 2015–2022, https://age-of-product.com/38-scrum-master-interview-questions-to-avoid-imposters-free-pdf.

> ***Tip:*** *Create interview teams of two teammates each—one asks questions while the other takes notes. After half of the interview has passed, they switch roles. The reason for switching is that most people need to improve at leading the conversation as well as at taking meaningful notes. Two people, however, have a much better chance than just one to recognize signals on the candidate's side, such as particular answers or body language.*

> ***Note:*** *The same procedure must apply to all candidates; otherwise, the results may not be comparable.*

It is a good practice to run these debriefings to aggregate the answers right after an interview with a candidate. Again, target for objectivity and have the Product Owner handle this task. They are professionals.

The most important question to answer is, "Would you like to work with the candidate?" And that question should be asked the following day. Sleeping on it will sober the interviewers and thus provide a path for a better decision.

> ***Tip:*** *Go with your first thought and walk away from any candidate you are not fully committed to in the morning. Don't rationalize your decision, as people can be taught new skills, but they won't change their personality. The trial day is an expensive exercise and should not be wasted.*

Candidates who are not considered for a trial day should receive an answer detailing the decision's reasons. I know that legal departments tend to freak out over this. They usually fear legal action, for example, on the grounds of anti-discrimination legislation. However, respect and transparency are essential values of the agile community and should be honored accordingly during the peer recruiting process.

Finally, invite the candidates that the Scrum Team would be interested in working with for a trial day. Let the team suggest a date, as they need to align a trial day with their Sprint rhythm.

7. Have a Trial Day

A trial day for a Scrum Master should not merely focus on basic Scrum mechanics. If you do that, you might risk choosing someone comfortable with "doing Agile by the book" (whatever book that is).

Hence, the purpose of the trial day is to get a practical understanding of how the future Scrum Master can support the whole organization in becoming (more) agile. The three top areas I focus trial days on are the Scrum Team, the product organization, and stakeholders and the organization beyond product and engineering:

A. The Scrum Team

This is the simple part. Here are some good exercises for hands-on learning:

- **Understanding the current state of the Scrum Team:** Have an introductory session with the complete team, a kind of "ask me anything" session for the candidate. A simple questionnaire will do the job; for example, 20 Questions a New Scrum Master Should Ask Their Team to Get Up to Speed.
- **Running a Retrospective:** Ask the candidate to run a Sprint Retrospective with the Scrum Team. Thirty minutes to prepare for the exercise should be sufficient. I would expect a seasoned Scrum Master to have prepared Retrospective formats at hand. (By "Retrospective," I am not referring to the basic "good, bad, and two actions items" 30-minute version. I would expect something a bit more sophisticated along the lines of Esther Darby and Diana Larsen's book, *Agile Retrospectives: Making Good Teams Great*.)[36]
- **Creating dashboards with agile metrics:** Visualizations are essential to stakeholder communication, and I believe the Scrum Master should take care of collecting data, aggregating information, and providing the gained knowledge in a way that helps the organization grow. In this exercise, the candidate is asked to create an initial version of such a dashboard and start collecting the first data. For example, following are typical metrics that are readily available by questionnaires or polls:
  - Team happiness
  - Perceived value delivered to customers during the last sprint
  - Perceived current level of technical debt

36. Esther Darby and Diana Larsen, *Agile Retrospectives: Making Good Teams Great* (Pragmatic Bookshelf, 2006).

- The Agile health level in the organization:
  - The Scrum checklist of Henrik Kniberg, which works fine for this purpose
  - Alternatively, the State of Agile Checklist for Your Organization

> ***Note:*** *Acquiring stakeholder feedback on the level of appreciation of the product delivery organization will most likely not be possible on a trial day.*

B. The Product Organization

Here are some good exercises for product organizations:

- **Understanding the current state of the Scrum Team from the Product Owner's perspective:** Have an interview with the Product Owner on the current situation from her perspective. Again, a simple questionnaire will do the job: 20 Questions from New Scrum Master to Product Owner. Suitable candidates will be prepared for that interview and will provide their ideas on how they can contribute to improving the agile product discovery and delivery process.
- **Participating in Product Backlog refinement:** The candidate should participate in a refinement session, helping the Scrum Team improve the Product Backlog in collaboration with the Product Owner—garbage in, garbage out, right? In addition, the candidate should demonstrate reasonable refinement practices during the exercise, addressing the following:
  - How to deal with large Product Backlogs
  - How to use the INVEST principle and the 3Cs to create Product Backlog items
  - How to handle acceptance criteria (e.g., use Gherkin)
  - The whole estimation vs. estimates process—estimation poker, knowledge transfer, #noestimates, predictability as an agile key metric

> ***Note:*** *A suitable candidate can ask the right question during the refinement session without having detailed knowledge about the Product Backlog. Handling the process and its principles are the focus of this exercise.*

C. The Stakeholders and the Organization beyond Product and Engineering

This part of the trial day assesses the future Scrum Master's communication capabilities. "Selling" the product and engineering organization to stakeholders and the

rest of the organization is valuable and essential to either further an agile transition or maintain its dynamic. It will be crucial in organizations with silos and legacy command-and-control structures outside of the product and engineering organization. This also applies to fast-growing startups with a lack of organizational structure, particularly when those are sales- or marketing-driven.

The task for the candidate will be to design a basic communication strategy with stakeholders suited to support transparency, interaction, and collaboration. (See also the section "Stakeholder Communication Tactics" later in this appendix.)

A worthwhile trial day usually requires a full working day and the attention of the whole Scrum Team, which is a pretty significant investment. So, choose the candidates carefully.

> ***Tip:*** *Invite the candidate—as well as Scrum Team members—for lunch. It will be impossible for them to play-act for an hour when interacting socially with several other people simultaneously. Having food together brings out true colors.*

> ***Note:*** *Menlo Innovations takes the trial process even further: "So we bring people in and get them to speed date with our own staff. The question is always: would you like to work with this person? If the answer is yes, then we bring them into work with us for a day, then a week, and then a month. If the answer is still, 'Yes, I would like to work with this person,' then they are hired."*[37]

8. Gather Feedback from the Team the Day after the Trial Day

Collect the feedback from the peer recruiting process from the team members the day after the trial day with a simple questionnaire:

How would you rate the candidate's competency level on a scale from:[37]

1 [Awesome!] to

6 [Thanks, but no thanks.]

Did the candidate do anything to impress you positively? (Free text field.)

Did the candidate do anything to impress you negatively? (Free text field.)

37. Steve Denning, "The Joy of Work: Menlo Innovations," *Forbes,* August 2, 2016, https://www.forbes.com/sites/stevedenning/2016/08/02/the-joy-of-work-menlo-innovations/#5791b4643069.

Would you consider working with the candidate as your new teammate?

Yes

No

Don't know

Should we make the candidate an offer?

Yes

No

Don't know

If the feedback is unanimous, it is the People Organization's task to take over. Either enter the contract negotiation or provide negative feedback from the Scrum Team and continue the search.

If the feedback is not unanimous, the team should discuss whether the differences are surmountable under the moderation from the People Organization. If they are insurmountable, the candidate should not be forced upon the Scrum Team. The team always has a veto right.

Conclusion

If your organization becomes agile, switching the hiring process to peer recruiting will be necessary. It won't make the Product Owner obsolete, but their role will change to facilitating others in choosing suitable candidates. Thus, the People Organization department will become a change agent, contributing to the agile transition of the organization.

On the other hand, sticking with the traditional command-and-control process would signal to everyone with an agile mindset that your organization isn't agile but is merely "doing Agile."

Retrospectives with Distributed Teams

The 5 Stages of a Retrospective

I modeled the following design of a remote Retrospective after the five stages of Esther Derby and Diana Larsen's book *Agile Retrospectives*[38]:

38. Derby and Larsen, *Agile Retrospectives*.

1. Setting the Stage

Impromptu Networking[39] is a simple application of breakout rooms; make sure that after each round, the pairs are created anew. Provide the invitation and the three questions—Why are you here? How can you contribute to making this a success for everyone? and What do you want to take home in return?—in advance.

Organizing a **Mad Tea** in the virtual realm requires a different approach. First, we cannot re-create two concentric circles of attendees facing each other. However, we can use the prompts—the half-sentences that the attendees shall complete—and the chat channel to create a quick and comprehensive picture of the team's sentiment. As the moderator, prepare a few prompts in advance regarding the topic of the remote Retrospective, for example, "When I think of our recent Sprint, I _____." Then post that prompt to the chat and ask the participants to add their answers but not to hit Enter. That is done simultaneously by all attendees when the host asks for it. The result is a bunch of responses on the same topic in the chat. Allow the team to scan the answers and move on to the second prompt.

Use check-ins with emojis to understand the sentiment of the attendees about the recent Sprint.

Choose from a growing list of online icebreakers. For example, see *Online Warm Ups & Energizers*[40] or *10 Fun Virtual Icebreakers to Take Remote Working to the Next Level*.[41] Consider crafting a working agreement for the upcoming meeting or workshop if the team hasn't yet done so.

2. Gathering Data

There are numerous ways to gather data for an upcoming Retrospective. For example, you probably want to track quantitative metrics such as cycle time or the number of bugs that escaped production. Or you might be interested in qualitative metrics such as team health or the sentiment of the team members. The point is that concerning the data, it does not matter whether the Scrum Team runs the

39. Lipmanowicz and McCandless, *The Surprising Powers*, 171–73.
40. Laïla von Alvensleben, "20+ Online Warm-ups & Energizers to Try with Your Team," September 1, 2022, https://mural.co/blog/online-warm-ups-energizers.
41. Anthony Murphy, "10 Fun Virtual Icebreakers to Take Remote Working to the Next Level," March 18, 2020, https://productcoalition.com/10-fun-virtual-icebreakers-to-take-remote-working-to-the-next-level-d764122e2e14.

analysis in a face-to-face or remote setting: both environments provide access to the same data. Following are typical practices of gathering data for Retrospectives:

- **Impromptu Networking** is an excellent way to create a sense of togetherness among the participants. It is also a valuable exercise to gather data if the invitation is crafted correctly.
- Anonymous surveys provide an option to collect data during the Retrospective as well as in advance. These surveys can be Sprint-specific, penciled in between the Sprint Review and the Retrospective. Alternatively, they can be open-ended surveys, such as a permanent suggestion box. Or they can be conducted regularly to track progress in areas of interest. For this purpose, suitable applications are Google Forms, Typeform, and SurveyMonkey. (I like to run anonymous surveys with Scrum Teams after each Sprint Review. Here, I ask for the perceived value created during the recent Sprint, the state of technical debt, the employer Net Promoter Score, and the personal sentiment: Are you happy, or are you looking for a new job?)
- A subset of the anonymous survey is the team radar. It is a great way to create transparency about important team matters and track their development as time passes. (One team radar I regularly run with all Scrum Teams, for example, is the Scrum Values radar.)
- Finally, you can derive metrics from supportive applications, for example, your ticket system. (Be careful, though, with the reports that are available out of the box. Often, they do not provide useful metrics specific to your situation.)

3. Generate Insight from the Data

After collecting the data, making sense of it is next. The following three Liberation Structures microstructures have proven to be useful, also in a remote setting:

- **What, So What, Now What?** is a sequence of individual and groupwork based on breakout rooms, aggregating findings in shared workspaces to be shared with the whole group.
- **TRIZ** combines essential elements of virtual Liberating Structures: breakout rooms, embedded 1-2-4-All, joined workspaces, and Shift & Share when several groups are working on the problem. Consider timekeeping via the breakout room broadcasting function, as participants are likely to be highly engaged and may lose track of time.

- Use the **Conversation Café**[42] by creating groups with the breakout room function, and identify a host for timekeeping. During rounds 1, 2, and 4, one participant is talking while the others are listening; use mute for the listeners. Once the timebox has expired, the previously talking participant hands over the microphone by calling out the next one in line and then muting themselves. As the facilitator, consider providing a matrix—rounds by speakers with checkboxes—to the hosts to ensure that everyone has a fair share of airtime.)

4. Deciding What to Do

The next step of the remote Retrospective is to agree on improvement items that will allow the team to grow and mature. Four Liberating Structures microstructures are well-suited for this purpose:

- **15% Solutions**[43] uses a procedure similar to that of TRIZ. Consider aggregating all suggestions in the whole group's shared workspace for clustering and ranking by voting.
- **25/10 Crowdsourcing**[44] allows for a ranked identification of actionable ideas to solve a problem. David Heath has created an excellent, free-to-use application[45] to run with distributed teams. When it comes to the often ensuing kerfuffle that occurs when running a 25/10 Crowdsourcing session with large groups in person, the application works remarkable stress-free.
- **Lean Coffee** is an excellent example of a workaround for virtual Liberating Structures. Gather all the input in the usual way, for example, engaging in 1-2-4-All, and put it on a board while voting is turned off. (Use several columns if the group is large to speed up the gathering process.) Then ask the entire group to cluster similar topics. Once the clustering is done, turn on the voting and order the remaining entries by votes. From here, continue with a group discussion or engage smaller groups with breakout rooms.
- **Ecocycle Planning**[46] applies the techniques from breakout rooms to shared workspaces. Speaking of which, given the large number of stickies that you usually create during Ecocycle planning, you may want to consider a specialized online

42. Lipmanowicz and McCandless, *The Surprising Powers,* 224–27.
43. Ibid., 191–93.
44. Ibid., 208–11.
45. David Heath, "25/10 Crowdsourcing Tool," https://twenty-five-ten.herokuapp.com.
46. Lipmanowicz and McCandless, *The Surprising Powers,* 295–99.

board application such as Miro or Mural. (Please note that neither tool is self-explanatory and requires a prep session with participants to avoid frustrating them.)

5. Closing the Retrospective

The last step of the Retrospective sequence is the closing or check-out. Basically, it is a mini-retrospective within the large remote Retrospective, focused on reflecting on what has happened and providing feedback: Was the time well spent, or do we need to change our approach to running a remote Retrospective next time? Keeping the process short and simple is paramount to enticing this feedback. Although we do not physically pass a door while leaving the meeting room, there are many ways to collect the feedback of the team members:

- We can replicate the door-sticky practice with the annotation tool of the video conferencing application on a prepared graphic. Then, all at the same time, attendees leave a symbol on a scale from 😡 to 😉 to 😀.
- Some applications allow for gathering live feedback, such as Poll Everywhere.
- Alternatively, run a Fist to Five voting.[47] (Ensure, though, that everyone knows the scale in advance. For example, include the scale in the working agreement if you use this voting practice regularly.)

SUPPORTING LIBERATING STRUCTURES

Two Liberating Structures can support a remote Retrospective with a larger team: 1-2-4-All and Shift & Share:

To cover 1-2-4-All, we need breakout rooms and a place to aggregate the findings. We start with everyone in the group for a minute in silence; then, we split the whole group into pairs using Zoom's or Team's breakroom feature for 2 minutes. After that round, we merge two pairs into a group of four for 5 minutes—this has to be done manually by the host—and the group aggregates its findings, for example, on a Google sheet prepared for each group in advance. Finally, we introduce each group's findings to the whole group by screen sharing in a Shift & Share.

In a Shift & Share, each workgroup presents its findings to the group through screen sharing. Alternatively, if the shared workspace has been created in advance,

47. Francois Favier, "SessionLab: Fist to Five," https://www.sessionlab.com/methods/fist-to-five.

for example, using Google Slides with a slide per workgroup, the moderator can share their screen while someone from the team explains the findings to the whole group. This moderator support reduces the stress of switching screen sharing on and off among several groups.

APPLICATIONS TO RUN A REMOTE RETROSPECTIVE

Of course, instead of tailoring a string of Liberating Structures to host a remote Retrospective, there are other options. Specialized Retrospective applications are available for distributed teams. Well-known providers are Retrium, FunRetro, and TeamRetro.

Then there are digital workspace applications, namely, digital whiteboards such as Miro and Mural, that often have prefabricated templates for Retrospectives. Another example is Atlassian's Confluence, which also provides a simple template that aligns with the Scrum Guide's purpose of a Retrospective.

Finally, workarounds based on general-purpose collaboration tools such as Google Workspace or Microsoft 365 also allow running at least rudimentary remote Retrospectives.

Good Practices for a Remote Retrospective

I want to point out three practices that make hosting significantly easier:

1. Create a script with the probable timeline of the remote Retrospective in advance. It shall include all the required documents to be shared with the participants and all the texts you must provide to the chat during the session.
2. Document the outcome of the remote Retrospective so team members can revisit them later. Restrict access to sensitive information by limiting access privileges strictly to team members. (Be prepared to explain the necessity of this procedure to curious or demanding line managers.)
3. Keep careful track of action items. Improvement items tend to be forgotten without a prominent placement in the team room. ("When you put problem in a computer, box hide answer. Problem must be visible!" Hideshi Yokoi, former President of the Toyota Production System Support Center in Erlanger, Kentucky, USA.)[48]

48. Hideshi Yokoi, quoted in "Problem Must Be Visible," Not Running a Hospital, August 10, 2009, http://runningahospital.blogspot.com/2009/08/problem-must-be-visible.html.

Conclusion

Working as a distributed agile team is, in many aspects, more complex than being co-located. Reading the room is significantly more complicated, for example, and communication takes a hit, as the beloved informal meeting at the coffee machine is no longer happening. However, being suddenly distributed does not mean we can no longer have useful critical events. On the contrary, the necessity to invest more preparatory work upfront may provide a chance to improve the meaning of events, and I would consider the remote Retrospective to be a prime candidate for that purpose.

Sprint Planning with Distributed Teams

Virtual Liberating Structures for a Remote Sprint Planning

To ensure that the physical barrier is not impeding a remote Sprint Planning, and similar to the remote Retrospective, we once again turn to virtual Liberating Structures to include and engage all Scrum Team members, ensuring that everyone has a voice.

For the remote Sprint Planning, the following Liberating Structures microstructures have proven to be supportive:

- The **Celebrity Interview**[49] is a valuable way for the Product Owner to introduce the upcoming Sprint's business objective. Working in the whole group, use mute and video off to distinguish between the Product Owner, the interviewer—a Developer—and the remaining Scrum Team members. Gather additional questions from the listeners through the chat channel as the interview proceeds. The interviewer should pass on these new questions in due time. (During the discussion, the Product Owner should not read these further questions simultaneously.)
- Use the **Conversation Café** by creating groups with the breakout room function, and identify a host for timekeeping. During rounds 1, 2, and 4, one participant talks while the others listen; use mute for the listeners. Once the timebox has expired, the previously talking participant hands over the microphone by calling

49. Lipmanowicz and McCandless, *The Surprising Powers*, 259–61.

out the next one in line and then muting themselves. As the facilitator, consider providing a matrix—rounds by speakers with checkboxes—to the hosts to ensure that everyone has a fair share of airtime.

- Consider running a **Mad Tea** to identify a Sprint Goal based on the business objective and the introduction of the Product Owner. Of course, we cannot re-create two concentric circles of attendees facing each other. However, we can use a prompt—a half-sentence that the team members shall complete—and the chat channel to create a quick picture of the team's sentiment on the Sprint Goal. As the moderator, prepare a few prompts in advance regarding the topic of the upcoming Sprint, for example, "I think the Sprint Goal should be _____." Then post that prompt to the chat and ask the participants to add their answer(s) but not to hit Enter. That is done simultaneously by all attendees when the host asks for it. The result is a bunch of suggestions on the Sprint Goal. Repeat the exercise if no suggestion finds unanimous support.
- Alternatively, **25/10 Crowd Sourcing** may prove helpful. This microstructure allows for a ranked identification of actionable ideas to solve a problem. Regarding the often ensuing kerfuffle when running a 25/10 Crowdsourcing session with large groups in person, the application works remarkably stress-free.
- **Min Specs** is an excellent exercise to focus the whole team on the essential work necessary to accomplish the next Sprint Goal. In a remote setting, it is a sequence of individual work and groupwork based on breakout rooms, aggregating findings in shared workspaces to be shared with the whole group in the end. Min Specs works well with embedded 1-2-4-All and Shift & Share.
- **Integrated~Autonomy**[50] helps to balance the demands of the Product Owner for new features with the necessity of the Developers to keep technical debt at bay, possibly already a latent conflict. During that session, ask, "How is it that we can be more integrated and more autonomous at the same time?" As Integrated~Autonomy builds on a sequence of small groupwork based on 1-2-4-All, breakout rooms, and a shared workspace to visualize the outcome, it is also well-suited for remote Sprint Planning.

SUPPORTING LIBERATING STRUCTURES

Three Liberating Structures can support a remote Sprint Planning with a larger team: 1-2-4-All, Shift & Share, and Lean Coffee.

50. Ibid., 287–90.

To cover **1-2-4-All**, we need breakout rooms and a place to aggregate the findings. We start with everyone in the group for a minute in silence; then, we split the whole group into pairs using Zoom's or Team's breakroom feature for 2 minutes. After that round, we merge two pairs into a group of four for 5 minutes—this has to be done manually by the host—and the group aggregates its findings, for example, on a Google sheet prepared for each group in advance. Finally, we can introduce each group's findings to the whole group by screen sharing in a Shift & Share.

In **Shift & Share**, each workgroup presents its findings to the group through screen sharing. Alternatively, if the shared workspace has been created in advance, for example, using Google Slides with a slide per workgroup, the moderator can share their screen while someone from the team explains the findings to the whole group. This moderator support reduces the stress of switching screen sharing on and off among several groups.

Lean Coffee is an excellent example of a workaround for virtual Liberating Structures. Gather all the input in the usual way, for example, engaging in 1-2-4-All, and gather it on a Mural or Miro board while voting is turned off. (Use several columns if the whole group is large to speed up the gathering process.) Then ask the entire group to cluster similar topics. Once the clustering is done, turn on the voting and order the remaining entries by votes. From here, continue with a group discussion or engage smaller groups with breakout rooms.

Conclusion

The task of aligning everyone becomes even more crucial for remote Sprint Planning than the planning of a co-located Scrum Team. The good news is that virtual Liberating Structures can help overcome this communication disadvantage of a distributed agile team by visualizing issues and giving every Scrum Team member a fair share of airtime.

Sprint Review with Distributed Teams

Virtual Liberating Structures for Remote Sprint Review

Sprint Reviews are crucial to fostering a shared understanding of a Scrum Team's progress toward the Sprint Goal. While challenging when meeting in person,

moving to a virtual environment often creates additional issues. However, applying Liberating Structures to remote Sprint Reviews can mitigate these challenges by including all participants and giving everyone a voice. Liberating Structures foster trust and help teams create value for customers, navigate competition, and overcome remote work challenges.

Virtual Sprint Review, Part 1: What Has the Scrum Team Accomplished during the Sprint?

Consider the following microstructures for the first phase of the remote Sprint Review:

- **Shift & Share** uses the science fair approach: members of the Scrum Team present different elements of the Increment to the whole group by screen sharing. (While Shift & Share properly includes the developers, the stakeholders might remain passive. Simple Ethnography, next, addresses that issue.)
- Using **Simple Ethnography,**[51] stakeholders explore the Increment on their own while the Scrum Team carefully observes what is happening. (If the stakeholders are thinking aloud during the exploration, it will improve the Scrum Team's capability to understand whether the Increment(s) are meeting expectations and where there is room for improvement. For this purpose, Simple Ethnography uses screen sharing and session recording for later analysis and is simple to move to a remote work environment.)
- User **Experience Fishbowl**[52] is an effective follow-up to Simple Ethnography. Working in the whole group, use mute and video off to distinguish between the inner circle—the stakeholders—and the outer circle—the Scrum Team members—of the User Experience fishbowl. Here, the outer circle members turn their video off and mute themselves. Next, gather additional questions from the outer circle members through the chat channel. A facilitator should pass on these new questions in due time. (While discussing a specific topic, the inner circle members should try not to read these new chat messages simultaneously.) Finally, debrief the whole group using What, So What, Now What?

51. Ibid., 283–86.
52. Ibid., 240–43.

Virtual Sprint Review, Part 2: Are We Still on Track to Accomplish the Product Goal, and Is Our Product Backlog Prepared for the Next Sprint?

Consider the following microstructures for the second phase of the remote Sprint Review:

- The **Celebrity Interview** is a valuable way for the Product Owner to repeat the upcoming Sprint's business objective and Product Goal, thus encouraging reflection on whether the Product Backlog is still up to its task. Working in the whole group, use mute and video off to distinguish between the Product Owner, the interviewer—a Developer or one of the stakeholders—and the remaining participants. Gather additional questions through the chat channel from the listeners as the interview proceeds. The interviewer should pass on these new questions in due time. (The interviewee should not read these further questions simultaneously during the discussion. The Celebrity Interview is also an excellent format to interview, for example, the CEO, when a shift in the organization's product strategy has occurred or is imminent.)
- **Min Specs** is an excellent exercise to focus the whole team on the essential work to accomplish the next Sprint Goal. In a remote setting, it is a sequence of individual work and groupwork based on breakout rooms, aggregating findings in shared workspaces to be shared with the whole group. Min Specs works well with embedded 1-2-4-All and Shift & Share.
- **Integrated~Autonomy** helps to balance the demands of the Product Owner for new features with the necessity of the Developers to reduce technical debt, possibly already a latent conflict. During that session, ask, "How is it that we can be more integrated and more autonomous at the same time?" As Integrated~Autonomy builds on a sequence of small groupwork based on 1-2-4-All, breakout rooms, and a shared workspace to visualize the outcome, it is also well-suited for remote Sprint Planning.

Virtual Sprint Review, Part 3: Closing

To close the remote Sprint Review, consider running a virtual **Mad Tea** to identify areas of improvement for the upcoming Sprint Retrospective. We cannot re-create two concentric circles of attendees facing each other. However, we can use a prompt—a half-sentence that the team members shall complete—and the chat channel to create a quick picture of the team's sentiment. As the moderator, prepare a few prompts in advance regarding the upcoming Sprint Review, for example,

"I think the next Sprint Review should be _____." Then post that prompt to the chat and ask the participants to add their answer(s) but not to hit Enter. That is done simultaneously by all attendees when the host asks for it. The result is a bunch of suggestions from the stakeholders and the Scrum Team members regarding the Sprint Review of the following Sprint, possibly also serving as data for the team's Retrospective.

Supporting Liberating Structures

Two Liberating Structures can support a remote Sprint Review with a larger team: **1-2-4-All** and **Lean Coffee**.

To cover **1-2-4-All**, we need breakout rooms and a place to aggregate the findings. We start with everyone in the group for a minute in silence; then, we split the whole group into pairs using Zoom's or Team's breakroom feature for 2 minutes. After that round, we merge two pairs into a group of four for 5 minutes—this has to be done manually by the host—and the group aggregates its findings, for example, on a Google sheet prepared for each group in advance. Finally, we can introduce each group's findings to the whole group by screen sharing in a Shift & Share.

Lean Coffee is an excellent example of a workaround for virtual Liberating Structures. Gather all the input in the usual way, for example, engaging in 1-2-4-All, and gather it on a Mural or Miro board while voting is turned off. (Use several columns if the whole group is large to speed up the gathering process.) Then ask the entire group to cluster similar topics. Once the clustering is done, turn on the voting and order the remaining entries by votes. From here, continue with a group discussion or engage smaller groups with breakout rooms.

Conclusion

Sprint Reviews are challenging at the best of times, and moving them into the virtual world compounds the difficulties. Given that the flow of information and the discovery of opportunities is likely to drop at the beginning of a Scrum Team's journey to remote work, the importance of the remote Sprint Review has increased significantly. Scrum Team, it is your turn now to overcome these impediments.

Stakeholder Communication Tactics

Stakeholder communication is not enough for an agile product development organization to create and ship the product like Swiss clockwork. It would help if you also talked about it, particularly at the beginning of your endeavor to become a learning organization. Marketing your journey to the rest of the organization—and thus securing their support, collaboration, and buy-in—is a critical success factor to step up the transformation game: you want to become agile, not do Agile.

Learn more about proven stakeholder communications tactics that contribute to making this happen.

Stakeholder Communication Channels

Deliver value to customers and talk about it—a simple necessity, mainly if your way of working as a Scrum Team is supposed to be embraced by the whole organization over time. Deciding early on how to communicate with internal stakeholders and win their support often makes the difference between doing Agile and becoming agile; see Figure B.14.

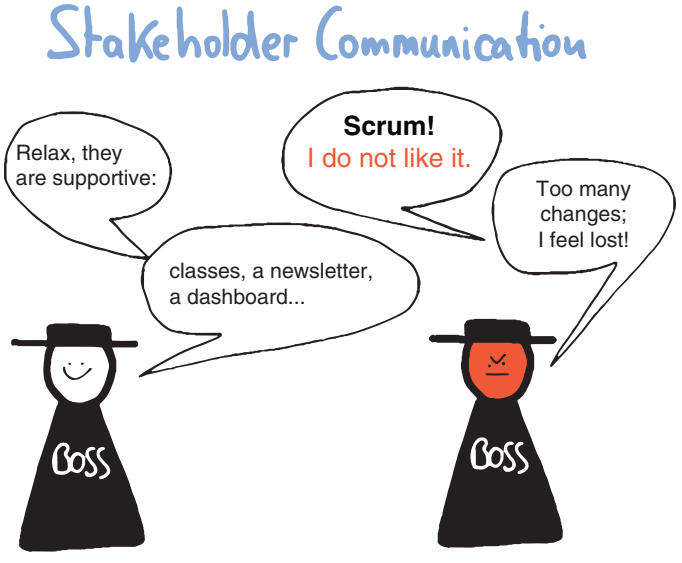

Figure B.14 Do good, talk about it, and include your stakeholders in the process.

In many organizations, executing to a plan is regarded as evidence of high performance. As a result, the status, compensation, and influence of people in the organization are tied, in one way or another, to executing "a plan." Merely telling people that creating plans and executing to those plans is bad and that you have a better way is likely to be met with derision and even hostility.

Run a quick mental exercise: try walking in your stakeholders' shoes, and ask yourself, "Would you entrust your career to some outrageous and unsupported claims by the protagonist of an agile way of working?"

Developing empathy for stakeholders is particularly relevant when the engineering and product development departments don't have the best standing within the organization at the beginning of an agile transformation. The good news is that no matter whether both departments are perceived as ineffective by others, a well-orchestrated communication strategy can contribute significantly to winning over the rest of the organization, provided the teams create value for customers.

Proven Communication Tactics

Judging by my experience, the following stakeholder communication tactics have proven helpful if actively pursued from the beginning. They all share a similar approach: they provide complete transparency to allow for inspection and adaptation by your stakeholders, fostering inclusion and giving everyone a voice.

Scrum Events

There are plenty of opportunities for stakeholders to interact responsibly with a Scrum Team. The two formal Scrum events and an adapted one that come to mind are Sprint Reviews, Daily Scrums, and stakeholder Retrospectives, respectively.

Sprint Reviews

The Sprint Review provides the opportunity for the Scrum Team and its stakeholders to inspect the Increment and the Scrum Team's progress toward the Product Goal. It also provides feedback that helps the team adapt the Product Backlog. The Developers, the Product Owner, and the stakeholders need to figure out whether they are still on track to deliver value to customers. It is the best moment to create or reaffirm the shared understanding among all participants whether the Product

Backlog still reflects the best use of the Scrum Team's time, thus maximizing customer value. It is because of this context that calling the Sprint Review a "sprint demo" does not match its importance for the effectiveness of the Scrum Team.

The cadence of this event depends on the Sprint length of your Scrum Teams, which is influenced, for example, by your product, your market, governance requirements, whether multiple Scrum Teams' Sprints are aligned, or whether they are practicing continuous delivery anyway.

A too-frequent Sprint Review, for example, every week, tends to wear off quickly, as only a little novelty can be provided. (The exception is if you've just started building something from scratch: then, a weekly Sprint Review is great for team building and alignment across the organization, thus creating a common spirit. It would help if you celebrated every victory.)

The basic rules of stakeholder communication with Sprint Reviews are simple:

- Educate the stakeholder on the importance of their collaboration.
- Lead by example. The Scrum Team(s) should be present. Why would stakeholders invest their time if the Scrum Team doesn't consider the Sprint Review a worthwhile investment of their time?
- Show, don't tell. Let the product increment speak for itself.
- Invite stakeholders to use the product increment.
- Keep it short and exciting. Using Liberating Structures to design Sprint Reviews has proven to be very helpful in including everyone and giving everyone a voice.

You may need to lure stakeholders to attend the Sprint Review if they don't yet fully understand the event's importance. Bribe them, if you have to: Accepting a few small tasks from those who attend works well. (But don't make this hack a habit.)

Of course, Sprint Reviews also work well with distributed teams in a virtual setting.[53]

53. Stefan Wolpers, "Remote Agile (Part 7): Sprint Review with Distributed Teams," April 27, 2020, https://age-of-product.com/remote-sprint-review-distributed-teams.

Daily Scrums

Invite stakeholders to the Daily Scrums to observe, once the Developers feel comfortable with the idea. You will need to manage assertive stakeholders who try to take over the Daily Scrum and turn it into a reporting session. But don't force the Scrum Team to let stakeholders attend if they don't want them to; after all, the Daily Scrum belongs to the Developers.

Stakeholder Retrospectives

Regularly—at least once a quarter—conduct a joint meta-level Retrospective that includes the stakeholders. A Meta-Retrospective is an excellent exercise to foster collaboration within the extended team, create a shared understanding of the big picture, and immediately create valuable action items. It comprises the team members of one or several product teams—or representatives from those—and the stakeholders. Participants from the stakeholder side are people from the business as well as customers. You find more information on Meta-Retrospectives in this Appendix.

STAKEHOLDER COMMUNICATION BY EDUCATIONAL INITIATIVES

Aggregate Information in Dashboards

As Hideshi Yokoi stated, "When you put problem in a computer, box hide answer. Problem must be visible!" In that spirit, be transparent with all information, but visualize it so stakeholders can make sense of it (often referred to in an agile context as *information radiators*).

Usually, it is more helpful to aggregate information across teams in a stakeholder dashboard than to have a board for each Scrum Team. Tracking the movement of subtasks, for example, usually proves too granular for stakeholders.

The advantages are plenty. Stakeholders rarely read reports, but they are willing to look at dashboards, and they appreciate a chat with Product Owners, Scrum Masters, and Developers. It is a controlled and safe environment, and you can choose the venue, too—put the dashboards where stakeholders can see them.

Be careful with simple burndown charts, though. Those easily get torn out of context and might cause a Pavlovian reflex with some stakeholders, triggering urges to micromanage teams.

Write Release Notes

Writing release notes improves stakeholder communication. Stakeholders often prefer them as they offer a concise summary of updates, saving time compared to self-navigation through a project management application, which may be less user-friendly or more time-consuming.

It helps to tell a story in your release notes. A story has a hero (your product, team, organization, etc.), a villain, an obstacle, and how the hero overcame the obstacle—in short: a story arc.

Invest some time to get the release notes right, and they will positively influence the standing of product and engineering within the organization. Rumor has it, they can even be fun to read. Further reading: "How to Write Release Notes (+5 Great Examples)."[54]

Institute Ambassadors

Start training interested individuals from the other departments—identified in a collaborative initiative with the respective department heads—to act as liaison officers to product and engineering. Ambassadors collect feedback, track feature requests, and check bugs before reporting.

Meet with them on a regular schedule, perhaps even weekly. They're excellent collaborators, often act as your proxies within their departments, and are well-liked user-test participants.

A note of caution: Don't bypass reluctant department heads just to avoid conflict.

Organize Training Classes

Invite other colleagues and stakeholders from the organization to regular training classes, and let them experience, hands-on, how your Scrum Teams build products using techniques like lean startup, user story mapping, versioning, prototyping—you name it. This kind of class does not include coding; prototypes are built with paper, crayons, pencils, and a prototyping app, for example, from InVision.

54. Anand Patel, "How to Write Release Notes (+5 Great Examples)," https://www.appcues.com/blog/release-notes-examples.

Read about lessons learned when working with marketing people, sales, and customer care agents here: "App Prototyping with Absolute Beginners."[55]

Stakeholder Communication by Regular Meetings

Offer Ask-Me-Anything Sessions in the Form of a Lean Coffee

Organize regular ask-me-anything sessions about agile practices, addressing your colleagues outside the product development organization: a "Lean Coffee is a structured but agenda-less meeting. Participants gather, build an agenda, and begin talking. Conversations are directed and productive because the agenda for the meeting was democratically generated."[56]

It is the ideal format to communicate your message and identify those interested in the approach in general. They will probably become valuable allies at a later stage.

Working Operationally in Stakeholder Departments

Everyone—Developers, product managers, Product Owners, and Scrum Masters—should regularly work in other parts of the organization, for example, in customer care. Nothing can create rapport more effectively than joining your colleagues in their operative work.[57]

With a modest investment of time, your Product Backlog item creation, and Product Backlog ordering process will become more aligned with solving real customer problems. Moreover, the whole product development organization will start rising from an anonymous group to valued colleagues.

---

55. Wolpers, "App Prototyping."
56. Lean Coffee, Lean Coffee Lives Here, http://leancoffee.org.
57. There is another advantage of joining customer care: "Customers are a big source of my e-mails. Anytime anyone contacts us with a question, whether it's by e-mail or telephone, they get a personal reply. The engineers and I handle customer support. When I tell people that, they look at me like I'm smoking crack. They say, 'Why would you pay an engineer $150,000 to answer phones when you could pay someone in Arizona $8 an hour?' If you make the engineers answer e-mails and phone calls from the customers, the second or third time they get the same question, they'll actually stop what they're doing and fix the code. Then we don't have those questions anymore." Learn more: Liz Welch, "The Way I Work: Paul English of Kayak," February 2, 2010, https://www.inc.com/magazine/20100201/the-way-i-work-paul-english-of-kayak.html.

Why? Because you served in their trenches, too. A user story or a bug report is no longer a ticket with a number in your ticket system; it is associated with a name, face, and story. Stakeholders will also learn more about your background and why solving something that looks trivial to them might cause a major engineering headache.

> ***Note:*** *I am not referring here to other regular stakeholder-level meetings, such as portfolio management, product roadmap planning, or user story mappings, as those would exceed the scope of this book.*

MEDIA CHANNELS

DAILY NEWSLETTER

Create a daily newsletter of five to six of the most important news pieces related to your company and industry. Throw in some startup- and technology-related posts on important trends or innovations, as well as an occasional article from product or engineering or the company blog.

Email list management tools will help you make sure the right people receive your updates, and their analytic capabilities will help you to gather feedback to improve the newsletter over time. (You can check popular links' click rates or run different headlines.)

Opening rates tend to be above 40 percent and can go higher if word gets around that the C level is actively reading the newsletter. Encourage people to provide you with interesting links and point them out to those contributors within the newsletter editions. Sometimes, you might be lucky and even start a friendly battle between individuals to provide valuable links. This exercise will not require more than 30 minutes per day.

> ***Side-effect:*** *The public relations department might need clarification about the intent of your newsletter to avoid the impression you are competing with them. (Depending on your organization's culture, you might even experience pushback.)*

PRODUCT AND ENGINEERING BLOG

Start blogging about your daily work to improve stakeholder communication: Technologies, processes, methodologies, frameworks, in other words: How you manage to ship a high-quality product, day after day. It significantly contributes to other components of the communication strategy sketched above if you can link to a detailed blog post.

A great example is Zalando's tech blog.[58] A product and engineering blog is also a cornerstone of a great recruiting strategy, going hand in hand with regular events, thus crossing the chasm from online to face-to-face communication with like-minded people you would like to hire.

CONCLUSION

If you fail at communicating your successes in creating value for customers to internal stakeholders, Agile might be regarded as a mere local process, probably a fad. Moreover, at the same time, the rest of the organization stays entrenched in silos and command-and-control structures. Once that state is reached, resisting any future agile improvement attempts—waving the banner of "we tried in the past, and it isn't working in our organization"—often becomes the prevailing attitude for those who are not supportive of agile and lean practices.

Resistance to change is the reason why considering proper stakeholder communication should be addressed on day one of your journey.

58. Zalando Engineering Blog, https://engineering.zalando.com.

Index

Numbers

1–2–4-All, 333
15% solutions, 352
20 questions
  from a new Scrum Master to the Developers, 312–315
  from a new Scrum Master to the Product Owner, 308–312
  from a new Scrum Master to the Scrum Team, 305–308
25/10 crowdsourcing, 352
100% in advance anti-pattern, 35, 203
360-degree feedback system, 7

A

absent Product Owner anti-pattern, 42–43
"acceptance" by the product owner anti-pattern, 51
acceptance criteria, Product Backlog, 206
additional work anti-pattern, 225–226
administrative assistant anti-pattern, 8–10
agile, 6–7
  anti-patterns, 5–6
  CoP (community of practice), 325
  metrics, 316
    bad, 322–323
    ugly, 323
  practices, cherry-picking, 88–89
  Scrum Master, 2
Agile Fluency game, 89
anonymous poll, 318–319
anti-pattern/s. *See also* remedy
  Definition of Done, 277
  Developer
    board out of date, 59–60
    cutting corners to meet an increment's deadline, 62–63
    daily status report, 70–71
    death by PowerPoint, 75, 165
    dispensable buffer, 76–77
    excessive feedback, 72
    fixed scope, 221–222
    gold-plating, 61–62, 113
    ignoring the Definition of Done, 278
    lack of professionalism, 58–59, 112
    lost Scrum Team, 68–69
    monologuing, 73, 147–149
    no capacity check, 66, 127–128

no impediments, 73–74, 149
No Routine, 68
no slack time, 64–66
no time for refinement, 77–78
no WiP limit, 57–58, 112
not supporting struggling developers, 73
planning meeting, 146–147
planning too detailed, 66, 220
problem-solving session, 71–72
same faces again, 75, 166–167
shadow Sprint Backlog, 224–225
showing off undone work, 75–76
side gigs, 112
submissive engineers, 79–81
technical debt, 63–64
too little planning, 66–67
too much estimating, 129, 220–221
unpreparedness, 69–70
incentivized stakeholder
financial incentives to innovate, 103–104
"My Budget" syndrome, 98–99
selling non-existent features, 101–103
"We know what to build", 100–101
IT management
all hands to the pumps without Scrum, 96–98
reassigning team members, 119–120
organization-level
line managers attend Retrospective, 192–193
no suitable venue, 191–192
Sprint success defined by output, not outcome, 267–268
Product Backlog
100% in advance, 203
everything is detailed and estimated, 205
INVEST?, 205–206
involving the Scrum team, 211–212
missing acceptance criteria, 206
outdated items, 204
oversized, 202–203
prioritization by proxy, 200–201
too detailed acceptance criteria, 206–207
Product Goal
Product Goal establishes how to accomplish the goal, 254–255
Product Goal is not transparent, 251
Product Owner forces the impossible upon the Scrum Team, 249–250
Product Owner singlehandedly creates Product Goals, 249
pursuing multiple, 248
there is no Product Goal, 247–248
Product Owner
100% in advance, 35
absent, 42–43
additional work, 225–226
assigning tasks to team members, 46–48
changing work items from the Sprint Backlog during the Sprint, 111
copy & paste, 32–33
crowding-out collaboration, 36–37
delaying, 44–45
"I know it all", 37, 163–164
introducing additional work, 48–50, 150
involving the Scrum team, 39–40
last minute changes, 133–134
last-minute Product Backlog, 39
no business objective, 271
no research, 38–39, 207
no Sprint cancellation, 45–46
output focus, 40–42
overreaching, 34–35
part-time owner, 31–32
prioritization by proxy, 35
Product Goal exclusionism, 251–252
Scrum team has no Sprint goal, 40
Sprint cancellations without consultation, 45
storage for ideas, 207–208
technical debt, 209–211
unfinished business, 132–133
what are we fighting for, 132
Scrum Master
acting as an agile manager, 5–6
administrative assistant, 8–10
Blame Game, 24, 189–191

Daily Scrum enforcer, 150–151
defining technical solutions, 18–19
excluding stakeholders all the time, 26
flow disruption, 17–18, 113
Groundhog Day, 22–23
no improvements, 17, 139
No Psychological Safety, 24–26
no slack time, 15–17
#NoDocumenatation, 24
#NoRetro, 24, 113
not preventing stakeholders from Daily Scrum attendance, 21–22
not supporting struggling developers, 20–21, 113
prisoners, 187–189
Retrospective postponement, 23
running Scrum events by allowing someone to speak, 7–8
spying for management, 11–13
team harmony over conflict resolution, 13–14
Scrum Team
changing Sprint Goals in the middle of a Sprint, 265
daily status report, 143
delivering X instead of Y, 237–238
disrespecting fellow team members, 144
dogmatism, 238
excessive feedback, 144–146
extensive complaining, 186–187
failure to achieve Sprint Goal on a regular basis, 264–265
forecast imposed, 137–138
handing the Increment over to QA, 234–235
"hardening" Sprint, 114–115
ignoring the purpose of the Sprint Review, 160–161
immediate user value is required in all Increments, 239–240
inability to accommodate work unrelated to Sprint Goal, 266–267
irregular Sprint lengths, 134–135
lost Scrum Team, 143
no Definition of Done, 281–282
no routine, 143–144
no Sprint Goal, 262
no Sprint Review, 159
no standard for "ready", 136–137
no visualization, 218–220
#NoAccountability, 184–185
#NoDocumentation, 186
not respecting Sprint boundaries, 265–266
overcrowding, 146
product Increment fails to achieve desired level of quality, 236–237
reintroducing waterfall planning through the backdoor, 253–254
rushed Retrospective, 180
skipping the Sprint Goal, 217
someone sings, 182–183
Sprint Goal imposed on Scrum Team, 262–263
Sprint Goal is confidential, 267
Sprint Goal is overly ambitious, 263–264
Sprint Zero, 115–117
sticking with obsolete Product Goals, 252–253
UNSMART action items, 184
variable Sprint length, 118
what improvement?, 185
stakeholder
cherry-picking individual practices, 88–89
command & control by line managers, 151–152
communicating via body language, 154
emergency work, 104
flow disruption, 105
internal stakeholders do not attend, 169–170
lack of leadership support, 86–87
lack of transparency, 84–86
no customers present, 170–172
Product Backlog and refinement, 105
root causes, 84
Scrum on a tight budget, 90–92
Sprint Review as approval gate, 167–169
starting over again, 172–173
steering committee meetings, 94

talkative, 152–154
telling people how to do things, 92–93
arbitrary deadlines, 62–63
ask-me-anything session, 366–367
autonomy, team, 98, 120–121, 188

B

bad agile metrics, 322–323
Baldauf, C., 23
bias
confirmation, 100
hindsight, 100
self-serving, 100
Blame Game anti-pattern, 24, 189–191
board out of date anti-pattern, 59–60, 112
body language, 106, 154
Bonaker, Z., WADE-matrix, 336
budget, 98–99
bypassing the product owner anti-pattern, 104

C

Celebrity Interview, 359
change, resistance to, 100–101. *See also* transformation
cherry-picking
individual practices anti-pattern, 88–89
Product Backlog items, 269–270
coaching
Developer, 74
one-to-one, 93
Scrum Master, 6–7, 14, 20
stakeholder, 169
collaboration, 93
in creating the Definition of Done, 283
between sales and Scrum teams, 102
command & control
culture, 117
by line managers anti-pattern, 151–152
communication, stakeholder
aggregate information in dashboards, 365
Daily Scrum, 364
Institute ambassadors, 366
release notes, 365–366
Sprint Reviews, 363–364
stakeholder Retrospective, 364
training classes, 366
confidentiality
Retrospective, 183
Sprint Goal, 267
confirmation bias, 100
conflict resolution, 13–14
continuous learning, Scrum Master, 6–7
Conversation Café, 352
CoP (community of practice), 324
overcoming resistance, 328
serving the community, 326
serving the organization, 327–328
copy & paste Product Owner, 32–33
creating
Definition of Done, 275–276, 334–335
Sprint Goals, 260–261
crowding-out collaboration anti-pattern, 36–37
customer/s
communication with Scrum Team, 95–96
feedback, 101, 229
cycle time, 57, 319–321

D

DAD (Discovery and Action Dialog (DAD), 333
daily newsletter, 367–368
Daily Scrum
communicating with stakeholders, 364
Developer anti-patterns
daily status report, 70–71
excessive feedback, 72
lost Scrum Team, 68–69
monologuing, 73, 147–149
no impediments, 73–74, 149
No Routine, 68
not supporting struggling developers, 73
planning meeting, 71, 146–147
problem-solving session, 71–72
unpreparedness, 69–70
Liberating Structures
1–2-4-All, 333
Lean Coffee, 333
Product Owner anti-patterns, introducing additional work, 150

purpose of, 142–143
remote, Liberating Structures
DAD (Discovery and Action Dialog (DAD), 333
Min Specs, 332
Troika Consulting, 330–332
WINFY (What I Need From You), 333
Scrum Master anti-patterns
Daily Scrum enforcer, 150–151
not preventing stakeholders from attendance, 21–22
not supporting struggling developers, 20–21
Scrum Team anti-patterns
daily status report, 143
disrespecting fellow team members, 144
excessive feedback, 144–146
lost Scrum Team, 143
no routine, 143–144
overcrowding, 146
stakeholder anti-patterns, 105–106
command & control by line managers, 151–152
communicating via body language, 154
talkative stakeholders, 152–154
daily status report anti-pattern, 70–71, 143
deadlines, arbitrary, 62–63
death by PowerPoint anti-pattern, 75, 165
defining technical solutions anti-pattern, 18–19
Definition of Done, 62, 235
acceptance by the Product Owner, 290
anti-patterns, 277
creating, 275–276, 334–335
Developer anti-patterns
ignoring the Definition of Done, 278
multiple versions of the Definition of Done in a single product, 279
Increments, 235–236
layers, 275–276
organization-level anti-patterns
dogmatism, 289–290
no common ground in a product group, 289
no transparency, 290
purpose of, 273–275
Scrum Team anti-patterns
Definition of Done is too demanding, 287
Increment fails to achieve desired level of quality, 284–285
Increment fails to meet the Sprint Goal, 285–286
Increments released that fail to meet the Definition of Done, 283–284
lack of collaboration in creating Definition of Done, 283
not improving the Definition of Done, 288
playing it safe, 286
Scrum Team has no Definition of Done, 281–282
too much ambition, too soon, 287–288
Definition of Ready, 279–281
delaying Product owner anti-pattern, 44–45, 111
Developer/s
coaching, 74
Daily Scrum anti-patterns, 68
daily status report, 70–71
excessive feedback, 72
lost Scrum Team, 68–69
monologuing, 73, 147–149
no impediments, 73–74, 149
No Routine, 68
not supporting struggling developers, 73
planning meeting, 71, 146–147
problem-solving session, 71–72
unpreparedness, 69–70
Definition of Done anti-patterns
ignoring the Definition of Done, 278
multiple versions in a single product, 279
Product Backlog anti-patterns
no time for refinement, 77–78
submissive engineers, 79–81
Retrospective anti-pattern, dispensable buffer, 76–77
role and responsibilities, 56
self-management, 47
Sprint anti-patterns
board out of date, 59–60, 112
cutting corners to meet an increment's deadline, 62–63

gold-plating, 61–62, 113
lack of professionalism, 58–59, 112
no slack time, 64–66
no WiP limit, 57–58, 112
side gigs, 60–61, 112
technical debt, 63–64
Sprint Backlog anti-patterns
fixed scope, 221–222
planning too detailed, 220
shadow Sprint Backlog, 224–225
too much estimating, 220–221
urge to deliver the complete Sprint Backlog, 222–223
Sprint Goal anti-patterns
cherry-picking Product Backlog items, 269–270
no visualization of progress, 270
Sprint Planning anti-patterns
no capacity check, 66, 127–128
no slack time, 129
planning too detailed, 66, 129
team leads, 68, 129–132
technical debt, 128–129
too little planning, 66–67
too much estimating, 129
Sprint Review anti-patterns
death by PowerPoint, 75, 165
same faces again, 75, 166–167
showing off undone work, 75–76
side gigs, 167
dispensable buffer anti-pattern, 76–77
disrespecting fellow team members anti-pattern, 144
dogmatism, 3, 238, 289–290

E

Ecocycle planning, 353
emergency work anti-pattern, 104, 121–122
excessive feedback anti-pattern, 72, 144–146
excluding stakeholders all the time anti-pattern, 26, 181–182
Experience Fishbowl, 359
extensive complaining anti-pattern, 186–187

F

failure to achieve Sprint goal anti-pattern, 113
features, nonexistent, selling, 101–103
feedback
360-degree, 7
customer, 101, 229
Developer, 149
excessive, 72
iterative, 104
late, 49
peer, 20, 149
peer recruiting team, 348–349
Product Owner, 44
sessions, 164
team, 72
fixed scope anti-pattern, 221–222
flawed tracking or useless metrics anti-pattern, 10–11
flow disruption anti-pattern, 17–18
Scrum Master, 113
stakeholder, 105
flow theory, 57
focus, 239
forecast imposed anti-pattern, 137–138
funding
agile transformation, 90–92
"My Budget" syndrome, 98–99

G

gold-plating anti-pattern, 61–62, 113
governance, 291
Groundhog Day anti-pattern, 22–23

H

"hardening" Sprint anti-pattern, 114–115
Hawthorne effect, 323
hindsight bias, 100
Hiring: 73 Scrum Master Interview Questions to Identify Suitable Candidates, 344

I

"I know it all" anti-pattern, 37
ignoring the Definition of Done anti-pattern, 278

impromptu networking, 349
incentivizing innovation, 103–104
Increment/s, 229–230
  Definition of Done, 283–284
  purpose of, 230–231
  Scrum Team anti-patterns
    all Increments must contribute to accomplishing the goal, 238–239
    delivering X instead of Y, 237–238
    dogmatism, 238
    failure to achieve desired level of quality, 284–285
    handing it over to QA, 234–235
    immediate user value is required in all Increments, 239–240
    no decision on the release of increments, 231–232
    only releasing to the complete user base, 233–234
    postponed releases, 232–233
    product Increment fails to achieve desired level of quality, 236–237
    released Increments not meeting the Definition of Done, 235–236
  stakeholder anti-patterns
    rebuilding legacy applications, 241–242
    roadmaps secrets, 240–241
inexperience, Scrum Master, 5
innovation, incentivizing, 103–104
Integrated~Autonomy, 359–360
internal stakeholders do not attend anti-pattern, 169–170
interviewing for Scrum Master position, 344–345
introducing additional work anti-pattern, 48–50, 150
INVEST principles, 205–206
involving the Scrum team anti-pattern, 39–40
irregular Sprint lengths anti-pattern, 134–135
IT management, Sprint anti-patterns
  all hands to the pumps without Scrum, 96–98, 118
  reassigning team members, 119–120
  team autonomy undermined, 120–121
iterative feedback, 104

J-K

Jeffries, R., 230
Jocham, R., 329
joint sales-Scrum charter, 102
Kanban board, 21, 72
Kniberg, H., 317

L

lack of leadership support anti-pattern, 86–87
lack of professionalism anti-pattern, 58–59, 112
lack of transparency anti-pattern, 84–86
last minute changes anti-pattern, 133–134
last-minute Product Backlog anti-pattern, 39
late feedback, 49
layers, Definition of Done, 275–276
laziness, Scrum Master, 2–3
lead time, 319–321
leadership
  Scrum Master, 2
  senior management, 86–87
Lean Coffee, 333, 352, 366–367
learning, workshops, 91
legacy applications, rebuilding, 241–242
Liberating Structure/s, 351–352, 353–354, 360–361
  1–2-4-All, 333
  DAD (Discovery and Action Dialog (DAD), 333
  Lean Coffee, 333
  Min Specs, 332
  remote Sprint Planning, 355–358
  TRIZ, 8–23
  Troika Consulting, 330–332
line managers, participation in Retrospective, 192–193
lost Scrum Team anti-pattern, 68–69, 143

M

management
  Product Backlog, 43
  self-, 47
  spying for, 11–13
media channels
  daily newsletter, 367–368
  product and engineering blog, 368

meetings, steering, 94
mentoring. *See also* coaching
  Product Owner, 164
  Scrum Master, 6–7
Meta-Retrospective, 335, 337–339
  how to run, 336
  Scrum Values exercise, 336–337
metric/s, 10–11, 40, 58, 101. *See also* agile
  agile, 316
    bad, 322–323
    good, 319–322
    ugly, 323
  cycle time, 319–321
  lead time, 319–321
  qualitative, 319
  self-assessment, 317–319
Min Specs, 332, 359
missing acceptance criteria anti-pattern, 206
monologuing anti-pattern, 73, 147–149
Musk, E., "The Secret Tesla Motors Master Plan", 85
"My Budget" syndrome anti-pattern, 98–99

N

new kid on the block anti-pattern, 117–118
no business objective anti-pattern, 271
no capacity check anti-pattern, 66, 127–128
no customers present anti-pattern, 170–172
no impediments anti-pattern, 73–74, 149
no improvements anti-pattern, 17, 139
no psychological safety anti-pattern, 24–26
no research anti-pattern, 38–39, 207
no routine anti-pattern, 143–144
no slack time anti-pattern, 129
  Developer, 64–66
  Scrum Master, 15
no Sprint Review anti-pattern, 159
no standard for "ready" anti-pattern, 136–137
no suitable venue anti-pattern, 191–192
no time for refinement anti-pattern, 77–78
no WiP limit anti-pattern, 57–58, 112
#NoAccountability anti-pattern, 184–185
#NoDocumentation anti-pattern, 24, 186
#NoRetro anti-pattern, 24, 113, 179
not supporting struggling developers anti-pattern, 73

O

one-to-one coaching, 93
organization-level anti-patterns
  Definition of Done
    dogmatism, 289–290
    no common ground in a product group, 289
    no transparency, 290
  Product Goal predetermines the how, 254–255
  Retrospective
    line manager attendance, 192–193
    no suitable venue, 191–192
    reporting structures within the Scrum Team, 194
  Sprint Goal
    Scrum Team lacks focus, 268
    Sprint success defined by output, not outcome, 267–268
  stakeholders dictate the Product Goal, 255–256
outdated Product Backlog items, 204
output focus anti-pattern, 134
overcrowding anti-pattern, 146
overreaching Product Owner anti-pattern, 34–35
oversized Product Backlog, 202–203

P

part-time owner anti-pattern, 31–32
passive stakeholders, 173–174
patience, 3, 7
peer feedback, 20, 149
peer recruiting, 342
  hiring a Scrum Master, 342–349
  People Operations and, 341–342
peer review, 104
People Operations, 341–342
pilot program, 92
planning meeting anti-pattern, 71, 146–147
planning too detailed anti-pattern, 66, 129, 220

Popper, K., 32
postponement
  increment release, 232–233
  Retrospective, 23, 180–181
practices. *See also* CoP (community of practice)
  agile, cherry-picking, 88–89
  remote Retrospective, 354
preparing, Sprint Planning, 126–127
prioritization by proxy anti-pattern, 35, 200–201
prisoners anti-pattern, 187–189
problem-solving session anti-pattern, 71–72
Product Backlog
  actionable, 199
  anti-patterns
    100% in advance, 203
    everything is detailed and estimated, 205
    INVEST?, 205–206
    missing acceptance criteria, 206
    outdated items, 204
    oversized, 202–203
    prioritization by proxy, 200–201
    too detailed acceptance criteria, 206–207
  Developer anti-patterns
    no time for refinement, 77–78
    submissive engineers, 79–81
  management, 43
  oversized, 35
  Product Owner anti-patterns
    last-minute Product Backlog, 207
    no research, 207
    storage for ideas, 207–208
    technical debt, 209–211
  purpose of, 198–200
  refinement session, 33
  Scrum Team anti-patterns
    involving the Scrum team, 211–212
    too much refinement, 212
  stakeholder anti-patterns, 105
Product Goal/s, 329
  anti-patterns
    Product Goal is not transparent, 251
    Product Owner forces the impossible upon the Scrum Team, 249–250
    Product Owner singlehandedly creates Product Goals, 249
    pursuing multiple Product Goals, 248
    there is no Product Goal, 247–248
  canvas, 330
  purpose of, 246–247
  Scrum Team anti-patterns
    exclusionism, 251–252
    reintroducing waterfall planning through the backdoor, 253–254
    sticking with obsolete Product Goals, 252–253
  stakeholder responsibility for creating and communicating, 255–256
Product Owner
  Daily Scrum anti-patterns
    assignments, 46–48
    introducing additional work, 48–50, 150
  Product Backlog and refinement anti-patterns
    100% in advance, 35
    copy & paste Product Owner, 32–33
    crowding-out collaboration, 36–37
    "I know it all" Product Owner, 37
    involving the Scrum team, 39–40
    last-minute Product Backlog, 39, 207
    no research, 38–39
    overreaching owner, 34–35
    oversized Product Backlog, 35
    part-time owner, 31–32
    prioritization by proxy, 35
    storage for ideas, 207–208
    technical debt, 209–211
  providing feedback, 44
  role and responsibilities, 30–31
  Sprint anti-patterns
    cancellations without consultation, 111
    changing work items from the Sprint Backlog, 111
    delaying, 111
    no Sprint cancellation, 112
    Sprint stuffing, 111
  Sprint Backlog anti-patterns
    additional work, 225–226
    changing items during the Sprint, 226–227

Sprint stuffing, 227–228
Sprint Goal anti-patterns, no business objective, 271
Sprint Planning anti-patterns
absent Product Owner, 42–43
delaying Product owner, 44–45
last minute changes, 133–134
last-minute changes, 40
no Sprint cancellation, 45–46
output focus, 40–42, 134
Scrum team has no Sprint goal, 40
Sprint cancellations without consultation, 45
unfinished business, 132–133
what are we fighting for, 132
Sprint Review anti-patterns
"acceptance" by the product owner, 51
"know-it-all" product owner loving their ideas, 51–52, 163–164
selfish owner, 50–51
product/s
discovery, 198
and engineering blog, 368
features, nonexistent, selling, 101–103
roadmap, 102, 240–241
psychological safety, 188, 191, 250
pull system, 59
purpose
of an agile community of practice, 325
of the Daily Scrum, 142–143
of Definition of Done, 273–275
of Increments, 230–231
of the Product Backlog, 198–200
of the Product Goal, 246–247
of the Retrospective, 178
of the Sprint, 109–110
of the Sprint Review, 158
pursuing multiple Product Goals, 248

Q-R

qualitative metrics, 319
quality/quality assurance, 234–235, 236–237, 283. *See also* Definition of Done
Increment, 284–285
radical candor, 148
reassigning team members anti-pattern, 119–120
rebuilding legacy applications, 241–242
refinement session, Product Backlog, 33
release notes, 365–366
remedy
absent Product Owner anti-pattern, 43
Developer anti-pattern
cherry-picking Product Backlog items, 269–270
cutting corners to meet an increment's deadline, 63
daily status report, 71
death by PowerPoint, 165
dispensable buffer, 76–77
fixed scope, 222
gold-plating, 62
ignoring the Definition of Done, 278
lack of professionalism, 59
lost Scrum Team, 69
monologuing, 148–149
no impediments, 74
no slack time, 65–66
no visualization of Sprint Goal progress, 270
no WiP limit, 57–58
planning meeting, 147
planning too detailed, 220
problem-solving session, 72
same faces again, 166–167
shadow Sprint Backlog, 225
showing off undone work, 76
side gigs, 61
submissive engineers, 79–81
team leads, 131–132
too little planning, 67
too much estimating, 221
unpreparedness, 70
urge to deliver the complete Sprint Backlog, 223
"hardening" Sprint anti-pattern, 115
incentivized stakeholder anti-pattern
financial incentives to innovate, 103–104
"My Budget" syndrome, 99

selling non-existent features, 102–103
"We know what to build", 101
IT management anti-pattern
reassigning team members, 119–120
team autonomy undermined, 120–121
new kid on the block anti-pattern, 118
organization-level anti-pattern
line managers participation in the Retrospective, 193
no suitable venue, 191–192
Sprint success defined by output, not outcome, 267–268
Product Backlog anti-pattern
100% in advance, 203
missing acceptance criteria, 206
outdated items, 204
oversized, 202–203
prioritization by proxy, 201
Product Goal anti-pattern
Product Goal is not transparent, 251
Product Owner forcing the impossible upon the Scrum Team, 250
Product Owner singlehandedly creates Product Goals, 249
pursuing multiple Product Goals, 248
there is no Product Goal, 247
Product Owner anti-pattern
"acceptance", 51
additional work, 226
assigning tasks to team members, 47–48
changing items during the Sprint, 227
copy & paste owner, 32–33
crowding-out collaboration, 36–37
delaying, 44
"I know it all", 37, 164
introducing additional work, 49–50
involving the Scrum team, 39–40
last minute changes, 134
last-minute Product Backlog, 39
no business objective, 271
no research, 38–39
no Sprint cancellation, 46
output focus, 42
overreaching owner, 34–35
part-time owner, 31–32
selfish owner, 50–51
storage for ideas, 207–208
technical debt, 211
unfinished business, 132–133
Scrum Master anti-pattern
administrative assistant, 9–10
"agile manager" behavior, 6–7
Blame Game, 190–191
Daily Scrum enforcer, 150–151
defining technical solutions, 19–20
flawed tracking or useless metrics, 11
flow disruption, 18
Groundhog Day, 23
irregular Sprint lengths, 135
no improvements, 17
No Psychological Safety, 24–26
no slack time, 16–17
not preventing stakeholders from Daily Scrum attendance, 22
not supporting struggling developers, 21
overly controlling Scrum Master, 8
prisoners, 188–189
Retrospective postponement, 23
spying for management, 12–13
team harmony over conflict resolution, 13–14
Scrum Team anti-pattern
Definition of Done is too demanding, 287
delivering X instead of Y, 237–238
disrespecting fellow team members, 144
excessive feedback, 144–146
excluding stakeholders all the time, 181–182
extensive complaining, 187
failure to achieve Sprint Goal on a regular basis, 264–265
forecast imposed, 137–138
ignoring the purpose of the Sprint Review, 160–161
inability to accommodate work unrelated to Sprint Goal, 266–267
Increments released that fail to meet the Definition of Done, 284
new kid on the block, 118

no decision on release of Increments, 232
no routine, 143–144
no Sprint Goal, 262
no Sprint Review, 159
no standard for "ready", 136–137
no visualization, 219–220
#NoAccountability, 185
#NoDocumentation, 186
#NoRetro anti-pattern, 179
not improving the Definition of Done, 288
only releasing Increments to the complete user base, 233–234
overcrowding, 146
overly ambitious Sprint Goal, 264
planning decisions made by definition of ready anti-pattern, 135
postponed Increment release, 233
Product Goal exclusionism, 252
released Increments not meeting the Definition of Done, 236
Retrospective postponement, 181
rushed Retrospective, 180
skipping the Sprint Goal, 217
someone sings, 183
Sprint Goal imposed on Scrum Team, 263
Sprint task accounting, 162
sticking with obsolete Product Goals, 253
unauthorized Sprint Backlog additions, 218
UNSMART action items, 184
variable Sprint length anti-pattern, 118
what improvement?, 185
Sprint stuffing, 228
stakeholder anti-pattern
command & control by line managers, 152
communicating via body language, 154
emergency work, 122
internal stakeholders do not attend, 169–170
lack of transparency, 85–86
no customers present, 171–172
passivity, 174
rebuilding legacy applications, 242
Scrum on a tight budget, 91–92
Sprint Review as approval gate, 168–169
starting over again, 172–173
talkative stakeholders, 153–154
talking to customers is off limits, 96
telling people how to do things, 93
remote Daily Scrum, Liberating Structures
1–2-4-All, 333
DAD (Discovery and Action Dialog (DAD), 333
Lean Coffee, 333
Min Specs, 332
Troika Consulting, 330–332
WINFY (What I Need From You), 333
remote Retrospective
applications for running, 354
closing, 353
deciding what to do, 352–353
gathering data, 350–351
generate insight from collected data, 351–352
good practices, 354
Liberating Structures, 353–354
setting the stage, 349–350
remote Sprint Planning, Liberating Structures, 355–358
remote Sprint Review
closing, 360
discussing accomplishments, 358–359
Liberating Structures, 358–361
reporting
-self, 12
structure within a Scrum Team, 194
Retromat.org, 23
Retrospective
Developer anti-patterns, dispensable buffer, 76–77
Meta-, 337–339
how to run, 336
Scrum Values exercise, 336–337
organization-level anti-patterns
line manager attendance, 192–193
no suitable venue, 191–192
reporting structures within the Scrum Team, 194
purpose of, 178
remote
applications for running, 354

closing, 353
deciding what to do, 352–353
gathering data, 350–351
generate insight from collected data, 351–352
good practices, 354
Liberating Structures, 353–354
setting the stage, 349–350
Scrum Master anti-patterns
Blame Game, 24, 189–191
Groundhog Day, 22–23
No Psychological Safety, 24–26
#NoDocumenatation, 24
postponement, 23
prisoners, 187–189
Scrum Team anti-patterns
excluding stakeholders all the time, 181–182
extensive complaining, 186–187
#NoAccountability, 184
#NoDocumentation, 186
#NoRetro, 179
postponement, 180–181
rushed Retrospective, 180
someone sings, 182–183
UNSMART action items, 184
what improvement?, 185
stakeholder, 364
stakeholder anti-patterns, 107
role and responsibilities
Developer, 56
Product Owner, 30–31
rushed Retrospective anti-pattern, 180

S

same faces again anti-pattern, 75, 166–167
Scrum, abandoning, 96–98
Scrum Master
43 ways to sabotage, 295–302
"agile manager" behavior, 6–7
anti-patterns
administrative assistant, 8–10
"agile manager" behavior, 5–6
flawed tracking or useless metrics, 10–11
overly-controlling, 7–8
spying for management, 11–13
team harmony over conflict resolution, 13–14
unrefined work items, 15–17
Daily Scrum anti-patterns
Daily Scrum enforcer, 150–151
not preventing stakeholders from Daily Scrum attendance, 21–22
not supporting struggling developers, 20–21
hiring, 342–349
mentoring and coaching, 6–7, 20
reasons for leaving the path, 4
dogmatism, 3
frustration, 3–4
impatience, 3
indifference, 3
inexperience, 5
laziness, 2–3
opportunism, 3
Retrospective anti-patterns
Blame Game, 24, 189–191
excluding stakeholders all the time, 26
Groundhog Day, 22–23
No Psychological Safety, 24–26
#NoDocumenatation, 24
postponement, 23
prisoners, 187–189
Sprint anti-patterns
defining technical solutions, 18–19
flow disruption, 17–18, 113
#NoRetro, 113
not supporting struggling developers, 113
Sprint Planning anti-patterns
no improvements, 17, 139
no slack time, 15
Scrum on a tight budget anti-pattern, 90–92
Scrum Team
autonomy, 188
communicating with customers, 95–96
Daily Scrum anti-patterns
daily status report, 143
disrespecting fellow team members, 144

excessive feedback, 144–146
lost Scrum Team, 143
no routine, 143–144
overcrowding, 146
Definition of Done anti-patterns
Definition of Done is too demanding, 287
Increment fails to achieve desired level of quality, 284–285
Increment fails to meet the Sprint Goal, 285–286
Increments released that fail to meet the Definition of Done, 283–284
lack of collaboration in creating Definition of Done, 283
not improving the Definition of Done, 288
playing it safe, 286
Scrum Team has no Definition of Done, 281–282
too much ambition, too soon, 287–288
Increment anti-patterns
all Increments must contribute to accomplishing the goal, 238–239
delivering X instead of Y, 237–238
handing the Increment over to QA, 234–235
immediate user value is required in all Increments, 239–240
no decision on their release, 231–232
only releasing to the complete user base, 233–234
postponed releases, 232–233
product Increment fails to achieve desired level of quality, 236–237
released Increments not meeting the Definition of Done, 235–236
leads, 129–132
output maximization, 40–42
Product Goal anti-patterns
exclusionism, 251–252
reintroducing waterfall planning through the backdoor, 253–254
sticking with obsolete Product Goals, 252–253
reassigning members, 119–120
reporting structure, 194
Retrospective anti-patterns
excluding stakeholders all the time, 181–182
extensive complaining, 186–187
#NoAccountability, 184–185
#NoDocumentation, 186
#NoRetro, 179
postponement, 180–181
rushed Retrospective, 180
someone sings, 182–183
UNSMART action items, 184
what improvement?, 185
Sprint anti-patterns
failure to achieve Sprint goal, 113
“hardening” Sprint, 114–115
new kid on the block, 117–118
Sprint Zero, 115–117
variable Sprint length, 118
Sprint Backlog anti-patterns
no visualization, 218–220
not having a Sprint Backlog by skipping the Sprint Goal, 217
unauthorized additions, 218
Sprint Goal anti-patterns
changing Sprint Goals in the middle of a Sprint, 265
failure to achieve Sprint Goal on a regular basis, 264–265
inability to accommodate work unrelated to Sprint Goal, 266–267
not respecting Sprint boundaries, 265–266
Scrum Team has no Sprint Goal, 262
Scrum Team lacks focus, 268
Sprint Goal imposed on Scrum Team, 262–263
Sprint Goal is overly ambitious, 263–264
Sprint Plan, 67
Sprint Planning anti-patterns
forecast imposed, 137–138
irregular Sprint lengths, 134–135
no standard for “ready”, 136–137
planning decisions made by definition of ready, 135
Sprint Review anti-patterns
ignoring the purpose of the Sprint Review, 160–161

no Sprint Review, 159
Sprint task accounting, 162
trial day, 346–348
self
-assessment, 317–319
-management, 47
-reflection, 20
-reporting, 12
-serving bias, 100
selling non-existent features anti-pattern, 101–103
senior management, leadership support, 86–87
shadow Sprint Backlog anti-pattern, 224–225
showing off undone work anti-pattern, 75–76
side gigs anti-pattern, 60–61, 112, 167
Simple Ethnography, 358
slack time, benefits, 65–66. *See also* no slack time anti-pattern
SMART framework, 184
someone sings anti-pattern, 182–183
Sprint
board, 58, 60
Developer anti-patterns
board out of date, 59–60, 112
cutting corners to meet an increment's deadline, 62–63
gold-plating, 61–62, 113
lack of professionalism, 58–59, 112
no slack time, 64–66
no WiP limit, 57–58, 112
side gigs, 60–61, 112
technical debt, 63–64
IT management anti-patterns
all hands to the pumps without Scrum, 118
reassigning team members, 119–120
team autonomy undermined, 120–121
Product Owner anti-patterns
cancellations without consultation, 111
changing work items from the Sprint Backlog, 111
delaying, 111
no Sprint cancellation, 112
purpose of, 109–110
Scrum Master anti-patterns
defining technical solutions, 18–19
flow disruption, 17–18, 113
#NoRetro, 113
not supporting struggling developers, 113
Scrum Team anti-patterns
failure to achieve Sprint goal, 113
"hardening" Sprint, 114–115
new kid on the block, 117–118
Sprint Zero, 115–117
variable Sprint length, 118
stakeholder anti-patterns
bypassing the product owner, 104
emergency work, 104
flow disruption, 105
stuffing, 111, 227–228
Sprint Backlog, 215–216
Developer anti-patterns
fixed scope, 221–222
planning too detailed, 220
shadow Sprint Backlog, 224–225
too much estimating, 220–221
urge to deliver the complete Sprint Backlog, 222–223
Product Owner anti-pattern
changing items during the Sprint, 226–227
Sprint stuffing, 227–228
Product Owner anti-patterns, additional work, 225–226
Scrum Team anti-patterns
no visualization, 218–220
not having a Sprint Backlog by skipping the Sprint Goal, 217
unauthorized additions to the Sprint Backlog, 218
Sprint Goal, 259–260
creating, 260–261
Developer anti-patterns
cherry-picking Product Backlog items, 269–270
no visualization of progress, 270
organization-level anti-patterns
Scrum Team lacks focus, 268
Sprint success defined by output, not outcome, 267–268

Product Owner anti-pattern, no business objective, 271
Scrum Team anti-patterns
changing Sprint Goals in the middle of a Sprint, 265
failure to achieve Sprint Goal on a regular basis, 264–265
not respecting Sprint boundaries, 265–266
Scrum Team cannot accommodate work unrelated to Sprint Goal, 266–267
Scrum Team has no Sprint Goal, 262
Sprint Goal imposed on Scrum Team, 262–263
Sprint Goal is confidential, 267
Sprint Goal is overly ambitious, 263–264
Sprint Planning
Developer anti-patterns
no capacity check, 66, 127–128
no slack time, 129
planning too detailed, 66, 129
team leads, 68, 129–132
technical debt, 128–129
too little planning, 66–67
too much estimating, 129
preparing, 126–127
Product Owner anti-patterns
absent Product Owner, 42–43
cancellations without consultation anti-pattern, 45
delaying, 44–45
last-minute changes, 40, 133–134
no Sprint cancellation, 45–46
output focus, 40–42, 134
Scrum team has no Sprint goal, 40
unfinished business, 132–133
what are we fighting for, 132
remote, Liberating Structures, 355–358
Scrum Master anti-patterns
no improvements, 17, 139
no slack time, 15
unrefined work items, 15–17
Scrum Team anti-patterns
forecast imposed, 137–138
irregular Sprint lengths, 134–135
no standard for "ready", 136–137
planning decisions made by definition of ready, 135
stakeholder anti-patterns, 106
Sprint Review
communicating with stakeholders, 363–364
Developer anti-patterns
death by PowerPoint, 75, 165
same faces again, 75, 166–167
showing off undone work, 75–76
side gigs, 167
Product Owner anti-patterns
"acceptance" by the product owner, 51
"know-it-all" product owner loving their ideas, 51–52, 163–164
selfish owner, 50–51
purpose of, 158
remote
closing, 360
discussing accomplishments, 358–359
Liberating Structures, 360–361
Scrum Team anti-patterns
ignoring the purpose of the Sprint Review, 160–161
no Sprint Review, 159
Sprint task accounting, 162
stakeholder anti-patterns, 106–107
bypassing the Product Owner, 122–123
emergency work, 121–122
internal stakeholders do not attend, 169–170
no customers present, 170–172
Sprint Review as approval gate, 167–169
starting over again, 172–173
Sprint Zero anti-pattern, 115–117
spying for management, 11–13
stakeholder/s
anti-patterns
cherry-picking individual practices, 88–89
financial incentives to innovate, 103–104
lack of leadership support, 86–87
lack of transparency, 84–86
"My Budget" syndrome, 98–99
root causes, 84
Scrum on a tight budget, 90–92

selling non-existent features, 101–103
steering committee meetings, 94
talking to customers is off limits, 95–96
telling people how to do things, 92–93
"We know what to build", 100–101
coaching, 169
collaboration with Scrum Team, 93
communication channels, 361–362
aggregate information in dashboards, 365
Daily Scrum, 364
Institute ambassadors, 366
release notes, 365–366
Sprint Review, 363–364
training classes, 366
Daily Scrum anti-patterns, 105–106
command & control by line managers, 151–152
communicating via body language, 154
talkative stakeholders, 152–154
excluding, 26, 181–182
Increment anti-patterns
rebuilding legacy applications, 241–242
roadmaps secrets, 240–241
passive, 173–174
Product Backlog anti-patterns, 105
responsibility for dictating the Product Goal, 255–256
Retrospective, 364
Retrospective anti-patterns, 107
Sprint anti-patterns
bypassing the product owner, 104
emergency work, 104
flow disruption, 105
Sprint Planning anti-patterns, 106
Sprint Review anti-patterns, 106–107
bypassing the Product Owner, 122–123
emergency work, 121–122
internal stakeholders do not attend, 169–170
no customers present, 170–172
Sprint Review as approval gate, 167–169
starting over again, 172–173
trust-building, 63
workshops, 101
steering committee meetings, 94
storage for ideas anti-pattern, 207–208
submissive engineers anti-pattern, 79–81

T

talkative stakeholders anti-pattern, 152–154
talking token, 149
team/s
autonomy, 98, 120–121
conflict resolution, 13–14
feedback, 72
review, 104
trust, 65
technical debt, 128–129, 233
Developer anti-pattern, 63–64
Product Owner anti-pattern, 209–211
telling people how to do things anti-pattern, 92–93
too little planning anti-pattern, 66–67
too much estimating anti-pattern, 129, 220–221
too much refinement anti-pattern, 212
training classes, 366
transformation, 89–90
funding, 90–92
leadership support, 86–87
pilot program, 92
transparency, 84–85. *See also* lack of transparency anti-pattern
Definition of Done, 290
Product Goal, 251
Sprint Goal, 267
TRIZ, 8–23
Troika Consulting, 330–332
trust, 12, 118
-building, 74
stakeholder, 63
team, 65, 93

U

ugly agile metrics, 323
unauthorized additions to the Sprint Backlog, 218
unfinished business anti-pattern, 132–133
unpreparedness anti-pattern, 69–70

unrefined work items anti-pattern, 15–17
UNSMART action items, 184

V

value, creation, 31
variable Sprint length anti-pattern, 118
Vegas rule, 182
visualization, 86
  Sprint Backlog, 218–220
  Sprint boards, 58
  Sprint Goal progress, 270

W-X-Y-Z

Wake, B., INVEST principles, 205–206
waterfall planning, 253–254
"We know what to build" anti-pattern, 100–101
what are we fighting for anti-pattern, 132
what improvement? anti-pattern, 185
WINFY (What I Need From You), 333
WiP (work-in-progress), 57
workshops, 91, 101